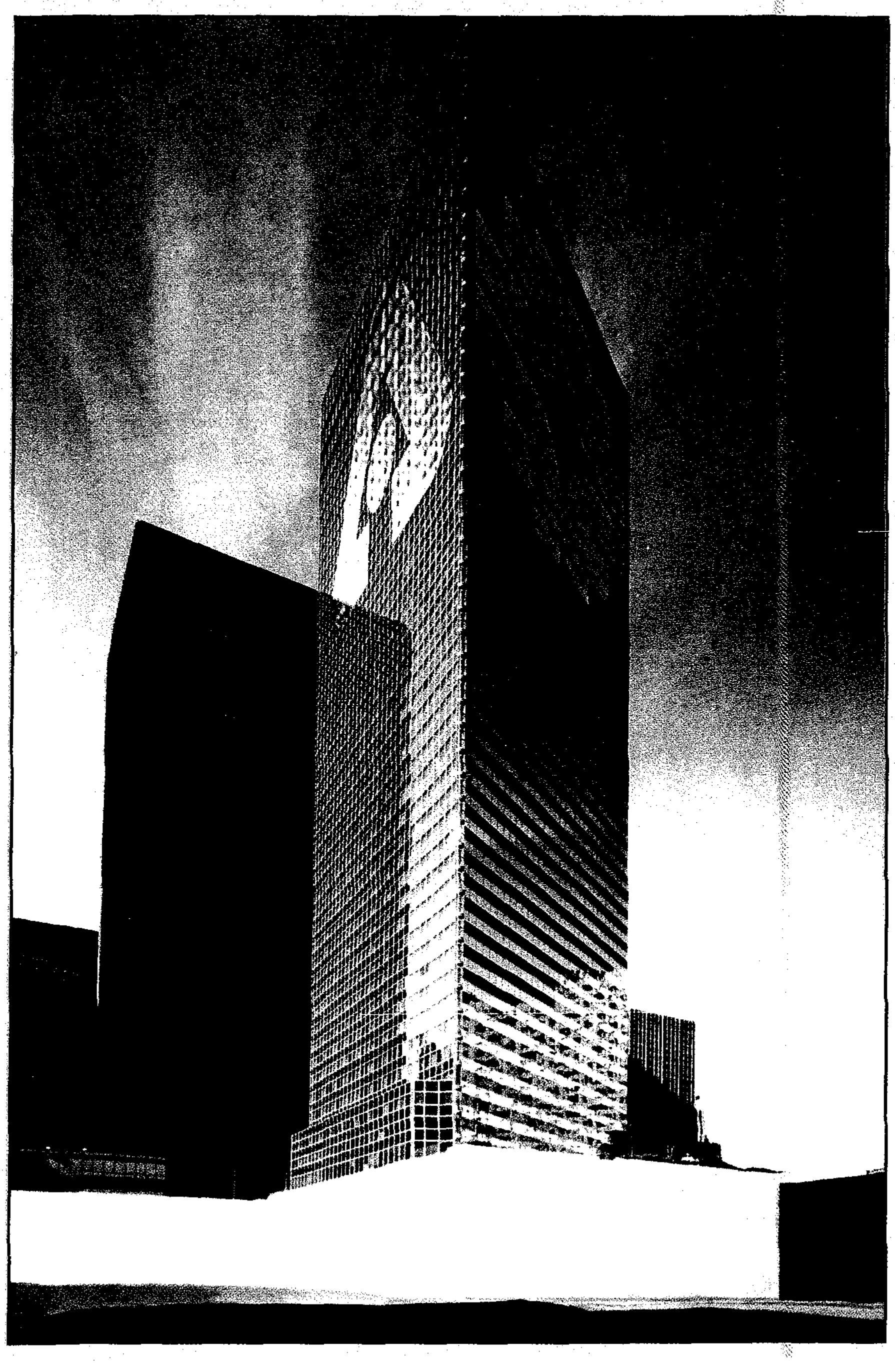
Proceedings of the 1995 International Conference on Accelerator and Large Experimental Physics Control Systems



Chicago, Illinois

October 30 - November 3, 1995

FERMILAB Conf-96/069

Proceedings of the 1995 International Conference on Accelerator and Large Experimental Physics Control Systems

Volume 1

Chicago, Illinois
October 30-November 3, 1995

Editors

M.C. Crowley-Milling
St. Cergue, Switzerland

P. Lucas
Fermi National Accelerator Laboratory
Batavia, Illinois

P. Schoessow
Argonne National Laboratory
Argonne, Illinois

Organized by

Argonne National Laboratory
Fermi National Accelerator Laboratory

Co-Organized by

EPCS EPS-Interdivisional Group for
Experimental Physics Control Systems

With Support from

The University of Chicago Board of Governors
for Argonne National Laboratory
URA Universities Research Association

Sponsored by

AAPPS Association of Asian Pacific Physical Societies
APS American Physical Society
DOE United States Department of Energy
EPS European Physical Society

In Cooperation with

IEEE-NPSS Nuclear and Plasma Sciences Society
of the IEEE

International Scientific Advisory Committee (ISAC)

Local Arrangements Committee

Coordinates

Chairman:	P. Lucas	Fermilab, MS 307
Executive Secretary:	L. Lopez	P.O. Box 500
		Batavia, IL 60510, USA

Phone:	1-708-840-4736
Fax:	1-708-840-8590
Telex:	373-6609
Internet:	icalepcs@almond.fnal.gov
World Wide Web:	http://www.aps.anl.gov/icalepcs95/icalepcs95.html

These proceedings are made possible in part due to grants from the U.S. Department of Energy and the Hewlett-Packard Corporation.

FERMILAB-Conf-96/069

PREFACE

Introduction

The fourth in a series of International Conferences on Accelerator and Large Experimental Physics Control Systems was held at the Swissôtel in Chicago from October 30 to November 3 1995. This series had been preceeded by a number of conferences on accelerator control systems dating to 1983, and it had later been decided to widen the scope to include other large experimental devices, such as telescopes, tokamaks and high energy physics experiments themselves, which have similar control problems.

The pre-conference announcements and circulars emphasized that certain aspects of the field would receive more stress at this conference than in the past. These areas were controls hardware; non-accelerator control systems, particularly those of physics experiments and telescopes; and operational aspects of controls. Oral sessions were devoted to each of these topics, two sessions in the case of hardware. Additionally, the topic of the opening plenary session talk was operational aspects. At a meeting held just after the conference closing the International Scientific Advisory Committee expressed the viewpoint that emphasis in these areas should be continued in the future. This will require particular effort in the areas of physics experiments and telescopes, where ICALEPCS is still not well known. At the same meeting the committee was divided on the value of continuing parallel sessions, which were held in Chicago for the first time.

Paper breakdown by topic

It is interesting to note the breakdown of the papers at this conference. The record 195 contained in these Proceedings have been categorized, including each paper no more than twice, as the following:

Accelerators	105
Physics experiments	17
Telescopes	5
Hardware	18
Operations	11
EPICS	12
Commercial and industrial	19
User interfaces	12
Distributed computing	12
AI and advanced techniques	9
OO techniques	11
Databases	7
Software sharing	3

The breadth of this distribution, as compared with that at early conferences, is notable and, speaking for the organizers, gratifying. A major new trend was noticed in the contributed papers - 36 of them could be classified as 'Control of subsystems.' A large fraction of these pertain to the extension of an existing control system to a new area, the upgrading of one part of an overall system, or control support for extension of machine functionality. The ability to make such changes indicates a major strength of the control system 'Standard Model', namely that it is modular and that additions or changes can be made in particular areas and then easily integrated into the overall system.

Commercial systems

One general impression of the progress presented at this conference is the greater readiness to adopt both hardware and software coming from outside the immediate community. There are several reasons for this. One is that the controls industry can now provide systems that meet the requirements for the control of many components, not just utilities, from their standard lines. Another is that many laboratories are suffering from reductions in staff and can no longer indulge in the luxury of in-house development.

Several papers show that such imports can be used with advantage for reducing the amount of in-house effort required in providing for the control of many of the support systems needed for an accelerator, but that they do have limitations. Many commercial control systems use proprietary networking and require their own separate networks. This may not create a major problem for small machines, but may involve extra expense for large ones, particularly if competitive tendering results in different suppliers for some of the subsystems. Commercial systems are not usually able to cope with the sub-millisecond timing required for instrumentation and RF.

Sharing of Software

As was the case at the 1993 ICALEPCS, a one-day workshop on the topic of sharing control system software was held after the completion of the conference. Compared with the first formal discussion at ICALEPCS 91, considerable progress in software sharing (SOSH) has been made. The EPICS collaboration, plus some smaller groupings of European laboratories, exists explicitly for this purpose. As is often pointed out, purchase of commercial software products as noted above is also a mode of sharing development costs. However some major laboratories, with large in-house-developed proprietary control systems, effectively cannot participate via such channels.

The conclusion of a previous workshop was that where sharing would have the most impact is in the matter of application software. There has been developed at CEBAF a product called CDEV which could serve as an interface between different application programming schemes on a high level and different control infrastructures and protocols on a lower plane. Several laboratories have investigated the possibility of such a product to interface their software to that of others and the results appear promising, although with obstacles. A major part of the Chicago workshop concerned specific detailed matters which would be involved in implementing such a scheme.

A separate effort involves sharing at a lower level in the controls model, namely a protocol at the level of the network or hardware bus serving similarly as a software bus. These discussions center primarily on the use of the CORBA method of object oriented communications. Such investigations continue.

On site computer center

One of the interesting features of the conference was the computer center which was constructed in the Swissôtel, primarily by members of the Fermilab Computer Division, for the use of all conference attendees and exhibitors. This center consisted of Macintosh and PC computers plus high end X-terminals, printers and a scanner. All were connected by an Ethernet LAN, with some wireless nodes attached. Wide area networking via a T1 link provided adequate bandwidth for the remote operation of various control systems. Some of the 'poster' presentations were actually made on X-terminals with outside connections. Additionally a projection system with connection to a computer or terminal at the podium allowed the visual aids of oral presentations to exist on computer screens, as opposed to transparencies. One of the hotel employees expressed surprise that for a technology conference this option was chosen by relatively few presenters. His remark can stand as a challenge to organizers of and presenters at future ICALEPCS.

Proceedings on CD-ROM

These Proceedings have been prepared in both paper and CD-ROM versions, 40% of the attendees having chosen the former and 60% the latter. It is clear that given the nature of the material we have received, CD-ROM is the more appropriate medium for this conference. Many authors have included detailed or colored figures and photographs, which are only approximated in the paper version. Additionally, the papers in a form appropriate for CD-ROM were also automatically appropriate for the World Wide Web, and thus were able to begin appearing in their final form within a month of conference completion.

Some comments on technical matters concerning the papers are appropriate. The instructions for authors sent before the conference stated rather explicitly how text was to be handled. As a result there were only minimal problems in preparing the text parts of papers for electronic media and with a consistency of format. However the same is not true of figures. These have been produced using a wide variety of software tools and in a variety of modes and formats. The result has been considerable work on our part to integrate all figures, and in a few cases with resulting quality and uniformity verging on inadequate.

Nomenclature

In the preface to the Proceedings of the last conference in this series, one of us pointed out that one of the remaining things that restricts the exchange of information between the laboratories and institutes is the plethora of names given to different parts of the systems.

The first that comes to mind is the names for the computers at the second or third layers of the standard model. We now have even more, not fewer, this time:

- DIC, Device Interface Computer
- DSC, Device Stub Controller
- ECA, Equipment Control Assembly
- EIU, Equipment Interface Units
- FEC, Front End Computer (Controller)
- FFE, Far Front End
- IDC, Intelligent Device Controller
- ILC, Intelligent Local Controller (Computer),
- IOC, Input/Output Controller,
- LPC, Local Process Controller (Computer) and
- PCA, Process Control Assembly.

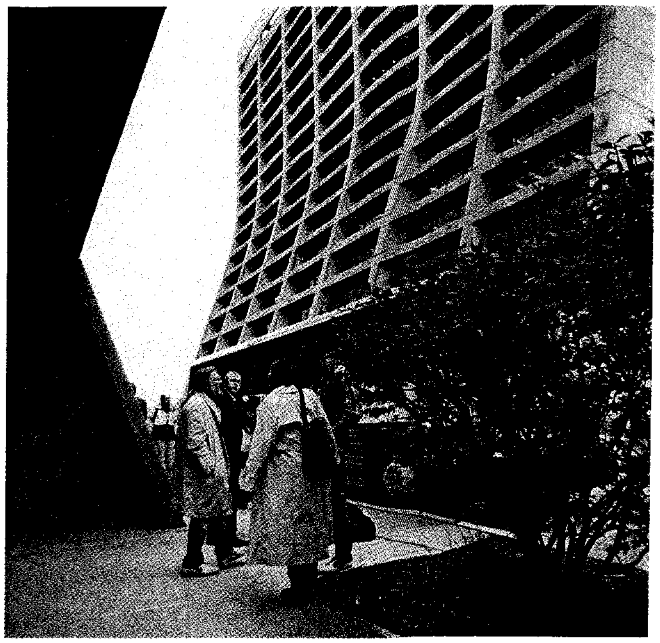
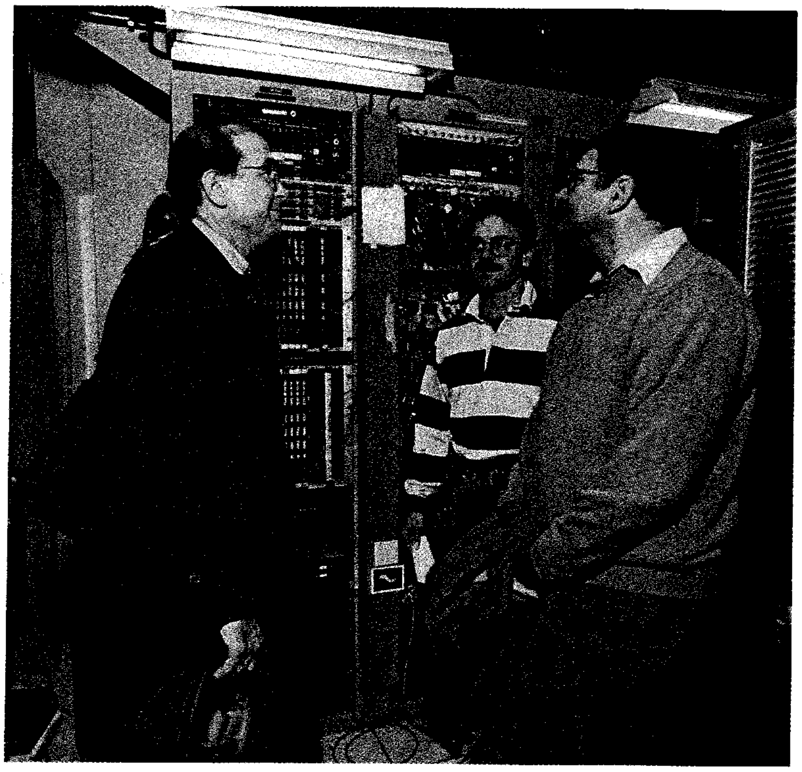
It would help if agreement could be obtained on a common name.

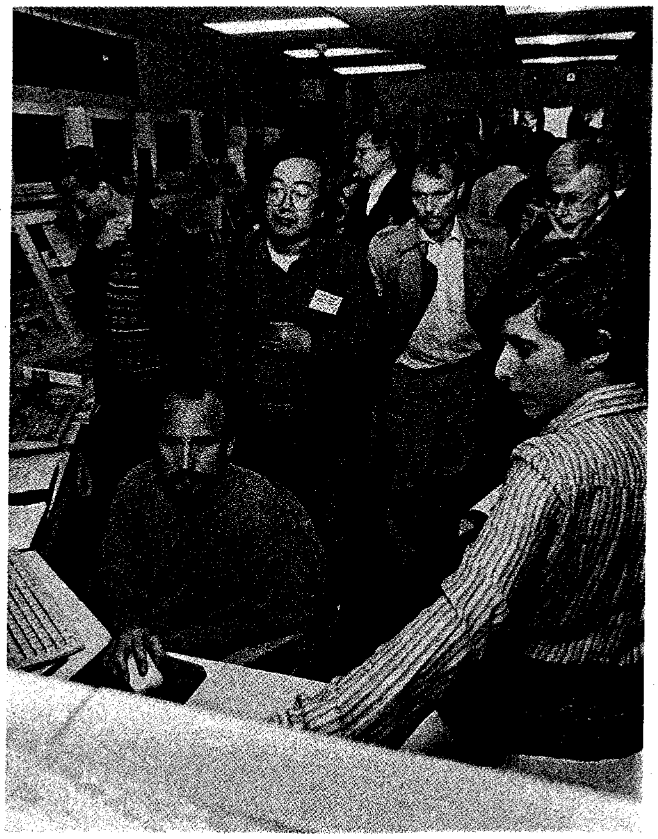
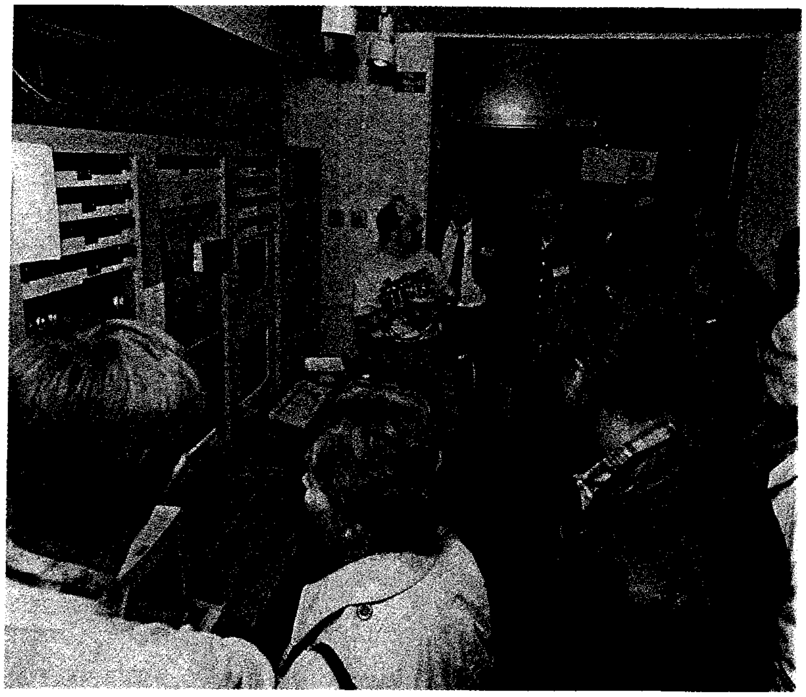
Acknowledgments

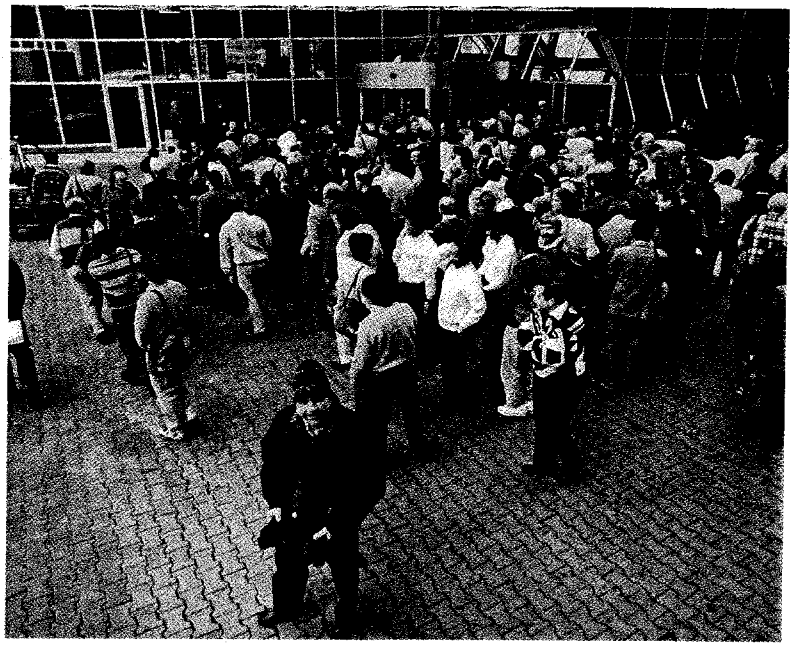
The smooth running of the conference depended on a number of people who devoted large amounts of time to ensuring its success. The names of the Local and International Committees can be found in the introductory pages. Also particularly noted are the efforts of Ms. Lisa Lopez who served as conference executive secretary and Ms. Donna Lamore who organized the construction and operation of the local computer center. We speak for all the organizers in expressing our hopes that all the conference participants enjoyed and profited from their visit to Chicago and that these Proceedings will be of value to them and to those who were unable to attend.

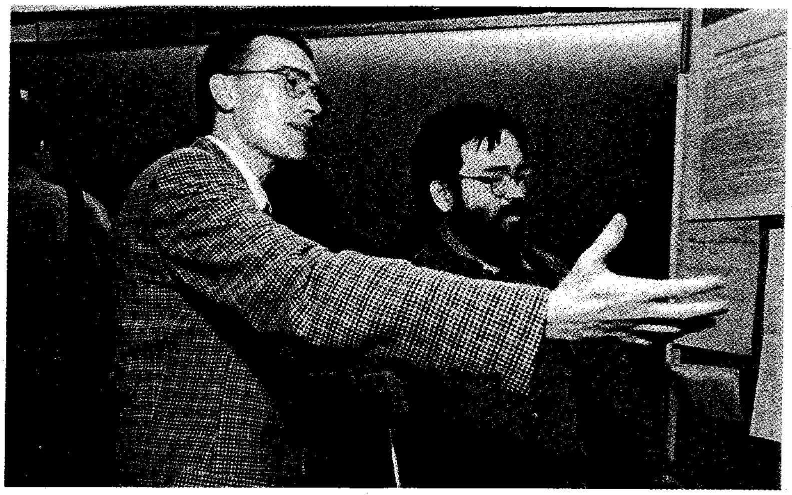
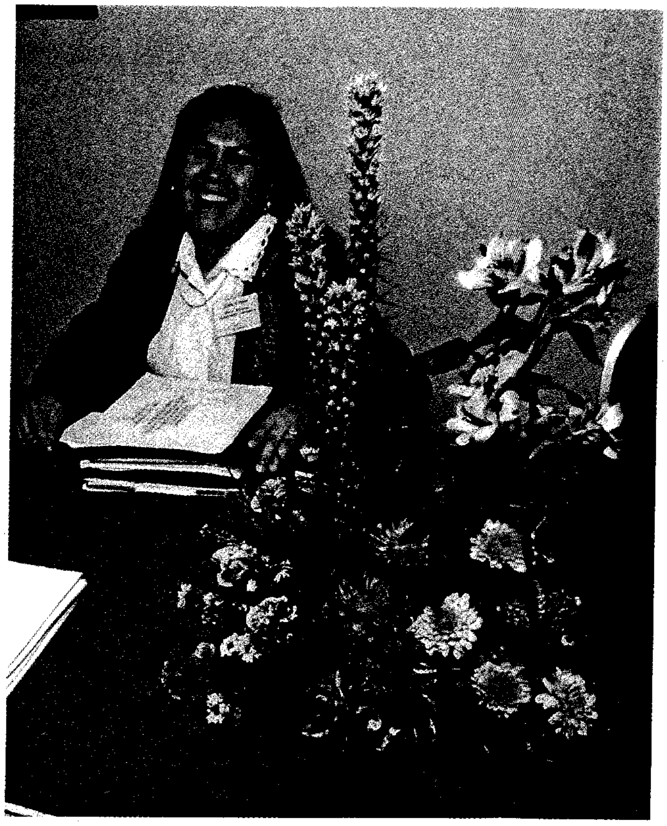
The production of the Proceedings was generously supported by the United States Department of Energy and the Hewlett-Packard Corporation. The organizers sincerely thank both organizations for their support.

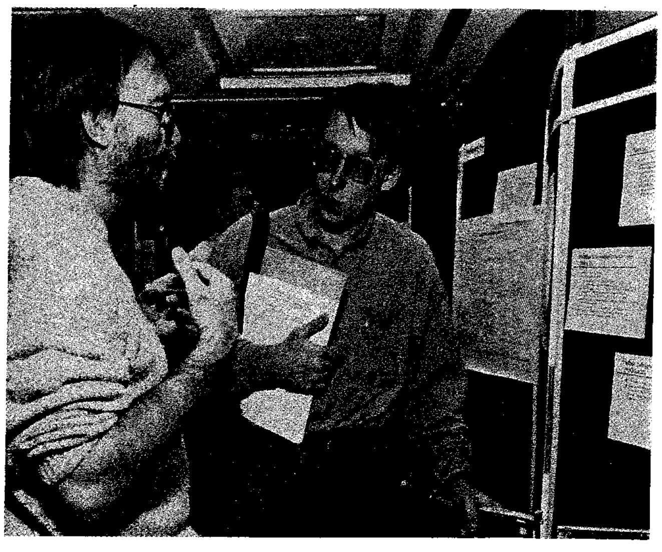
Michael Crowley-Milling
Peter Lucas
Paul Schoessow

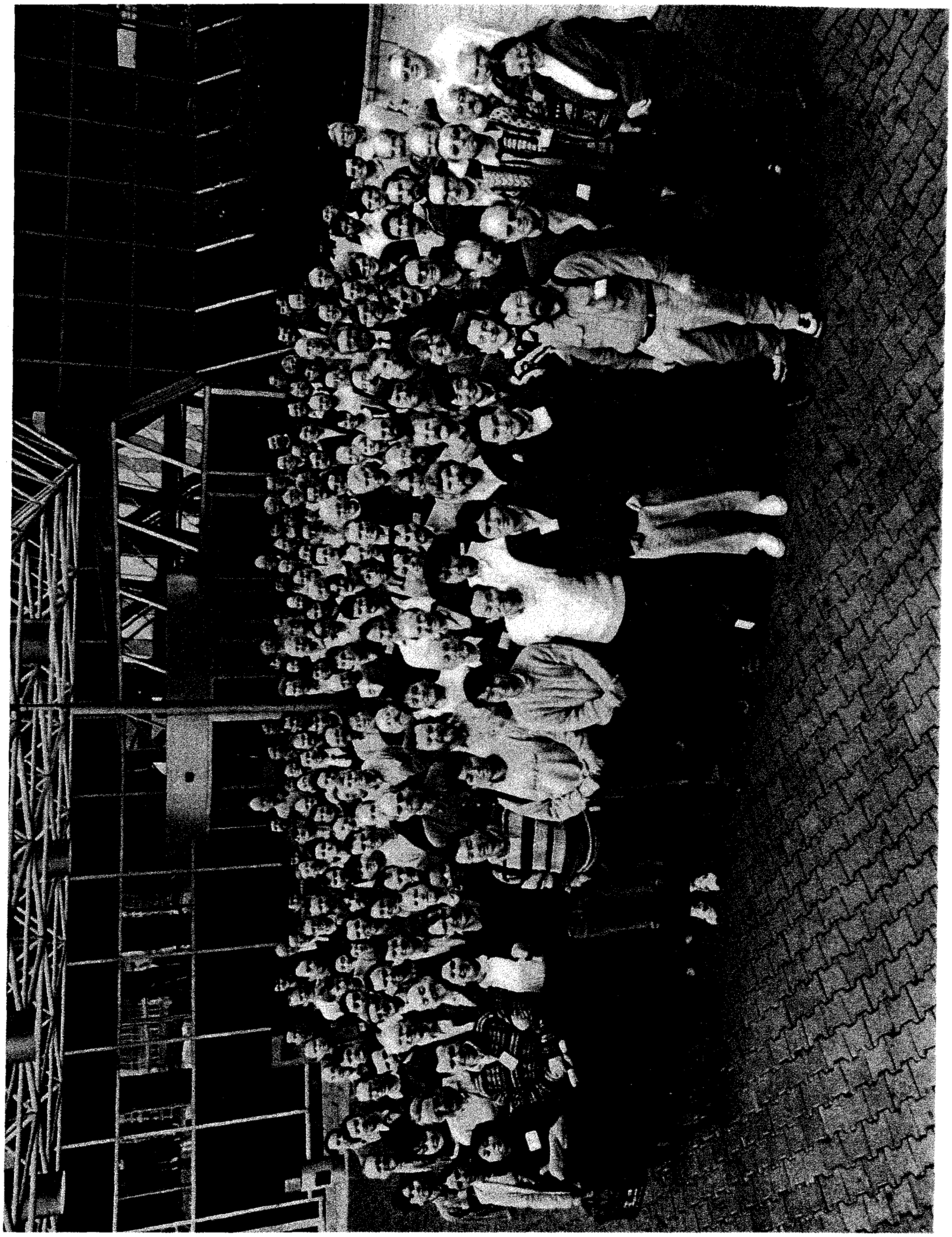



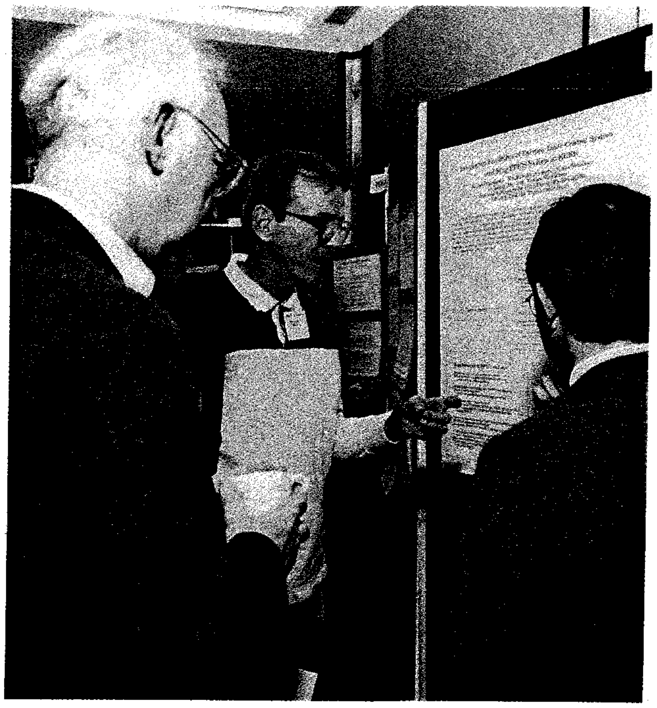
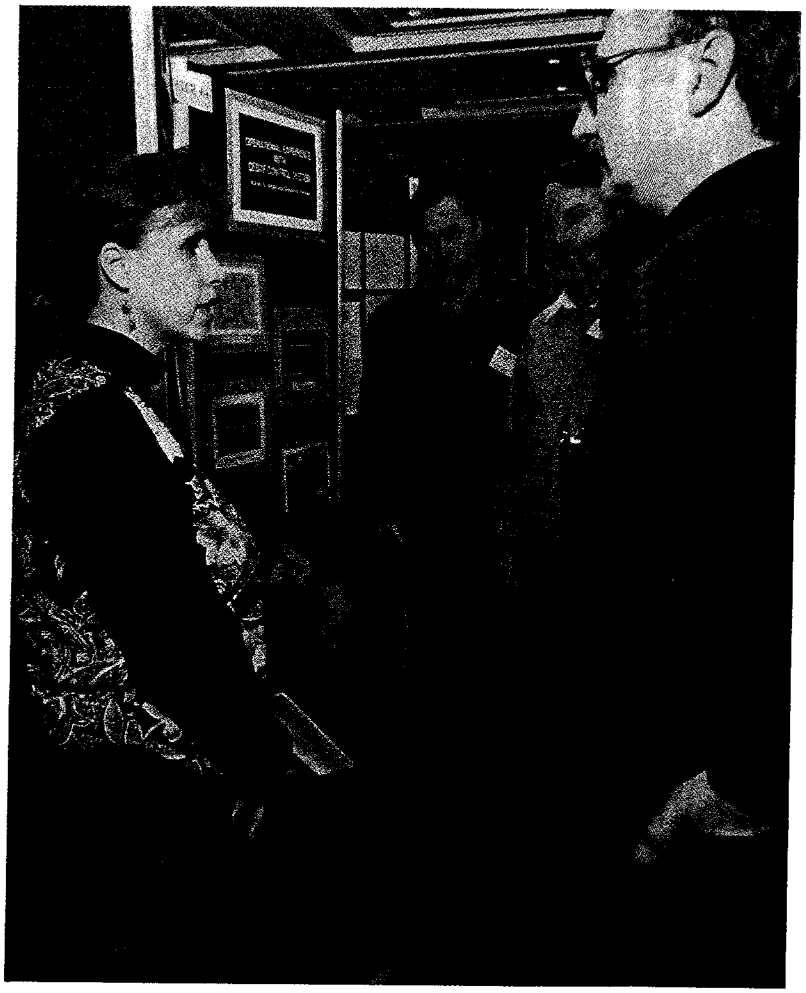
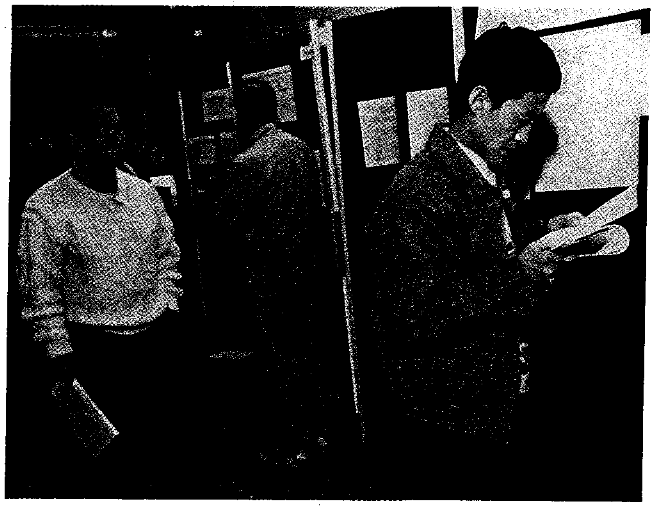
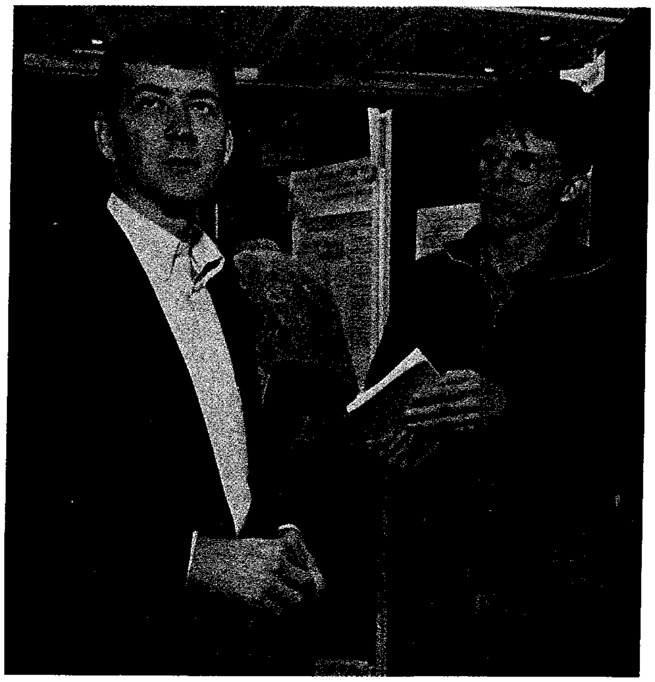
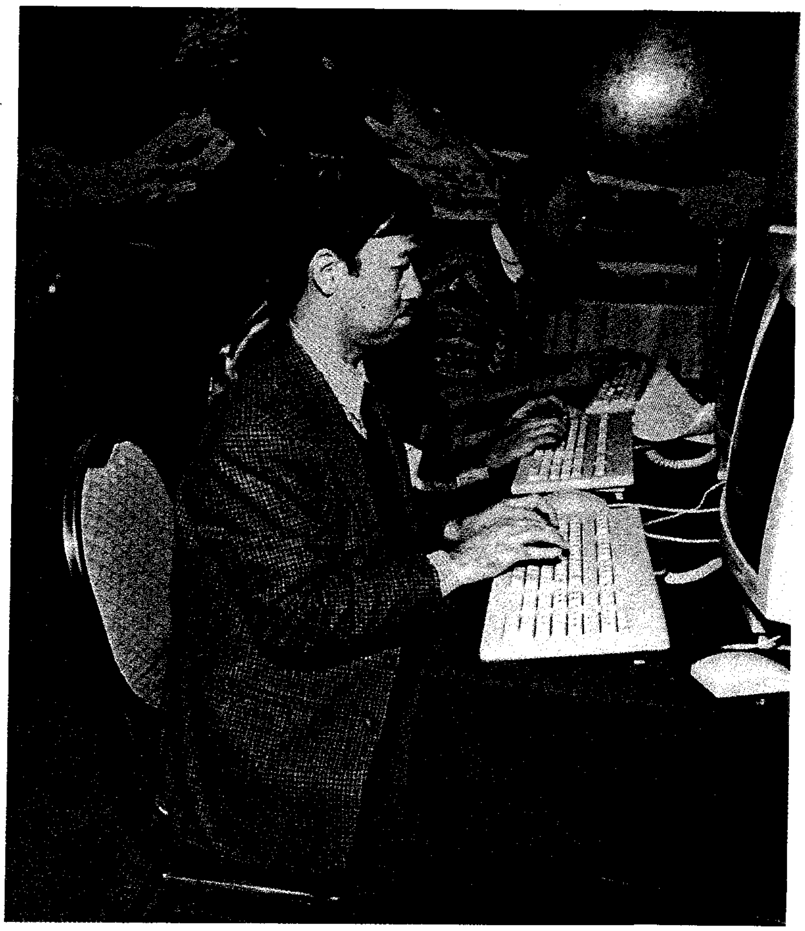









List of Participants

Abe	Isamu	KEK
Abola	Agnes C.	Brookhaven National Laboratory
Adhikari	Jiwan Singh	Centre for Advanced Technology-Indore
Ahn	Seung-chan	Fermi National Accelerator Laboratory
Albrand	Solveig	Institut des Sciences Nucleares
Alferov	Vladimir N.	Institute for High Energy Physics
Allison	Stephanie A.	SLAC
Anderson	Janet B.	Argonne National Laboratory
Anicic	Damir	Paul Scherrer Institute
Arnold	Ned D.	Argonne National Laboratory
Arruat	Michel	CERN
Bailey	Roger	CERN
Bakken	Jon A.	Fermi National Accelerator Laboratory
Banerjee	Bakul	Argonne National Laboratory
Baribaud	Guy M.	CERN
Bartlett	Frederick J.	Fermi National Accelerator Laboratory
Barton	Donald S.	Brookhaven National Laboratory
Bickley	Matthew M.	CEBAF
Billen	Ronald R.	CERN
Biocca	Alan K.	Lawrence Berkeley National Laboratory
Birke	Thomas	BESSY
Blokland	Willem	Fermi National Accelerator Laboratory
Bluem	Hans P.	Louisiana State University
Blumer	Thomas H.	Paul Scherrer Institut
Boriskin	Victor N.	Kharkov Institute of Physics and Technology
Borland	Michael	Argonne National Laboratory
Boutheon	Marcel A.M.	CERN
Bowling	Bruce A.	CEBAF
Bowling	P. Stuart	Los Alamos National Laboratory
Bozoki	Eva S.	Brookhaven National Laboratory
Bracey	Mark	SDS, Inc.
Brazier	John C.L.	Brazier Systems & Consultants
Briegel	Charles I.	Fermi National Accelerator Laboratory
Brown	Gregory L.	Fermi National Accelerator Laboratory
Brown	Stanley K.	Los Alamos National Laboratory
Bulfone	Daniele	Sincrotrone Trieste
Burckhart	Helfried J.	CERN
Busse	Winfried W.	Hahn-Meitner-Institut
Cahill	Kevin J.	Fermi National Accelerator Laboratory
Carlier	Etienne	CERN
Carmichael	Linden R.	Fermi National Accelerator Laboratory
Carroll	Thomas J.	University of Richmond
Cha	Ben-Chin K.	Argonne National Laboratory
Chaize	Jean-Michel	ESRF
Charrue	Pierre	CERN
Chase	Brian E.	Fermi National Accelerator Laboratory
Chen	Jie	CEBAF
Chepurnov	Alexandre S.	Moscow State University
Chiozzi	Gianluca	European Southern Observatory
Chowdhary	Mahesh	CEBAF
Chung	Youngjoo	Argonne National Laboratory

Ciapala	Edmond	CERN
Ciarlette	Daniel J.	Argonne National Laboratory
Ciriani	Paolo	CERN
Clausen	Matthias	DESY
Clayton	Michael J.	CERN
Coleman	Thomas A.	Argonne National Laboratory
Collins	Bill	Dawn VME Products
Collins	John C.	Indiana University-CF
Copely	Tony	Tektronix, Inc.
Craft	Benjamin C.	Louisiana State University-CAMD
Dale	Donald J.	TRIUMF
Dalesio	Leo R.	Los Alamos National Laboratory
Daley	Dan	Hewlett Packard Corp.
Daly	Robert R.	Argonne National Laboratory
Dambik	Edward J.	Fermi National Accelerator Laboratory
de Vries	Jannes C.	CERN
Dean	Thomas D.	SLAC
Denis	Bernard M.	CERN
Di Maio	Franck	CERN
Dinius	Arend	CERN
DiPirro	Giampiero	INFN-LNF Frascati
Dohan	Donald A.	Paul Scherrer Institute
Douglas	Greg	GMW Associates
Dunham	Bruce M.	CEBAF
Duval	Philip D.	DESY-MKI
Eisert	David E.	Synchrotron Radiation Center
Engstršm	Mats	Manne Siegbahn Laboratory
Epaud	Francis	ESRF
Evans	Ken	Argonne National Laboratory
Feng-Berman	Shuchen K.	Brookhaven National Laboratory
Feres	Jacques	MIDI Ingenierie
Firebaugh	Jerry D.	Fermi National Accelerator Laboratory
Flannigan	John	Brookhaven National Labortory
Florian	Robert V.	Fermi National Accelerator Laboratory
Forrestal	John R.	Argonne National Laboratory
Foth	Lynn M.	Michigan State University
Fox	John D.	SLAC
Frammery	Bertrand G.	CERN
Franck	Al R.	Fermi National Accelerator Laboratory
Friedman	Nick	Argonne National Laboratory
Fuess	Stuart C.	Fermi National Accelerator Division
Fullett	Kenneth D.	Fermi National Accelerator Laboratory
Furukawa	Kazuro	KEK
Gajewski	Konrad	The Svedberg Laboratory
Garrett	Richard A.	SLAC/SSRL
Gavaggio	Richard F.	CERN
Gilot	Jean-Francois	Creative Electronic Systems
Goodwin	Robert W.	Fermi National Accelerator Laboratory
Gougnaud	Francoise	CEA/Saclay
Gournay	Jean-Francois	CEA/Saclay
Guerrero	Louis	CERN
Guiard-Marigny	Alain	CERN
Gurd	David P.	Los Alamos National Laboratory
Halling	Alfred	Fermi National Accelerator Laboratory

Harms	Michael G.A.	SLAC
Harrison	James F.	Los Alamos National Laboratory
Hart	Robert G.K.	NIKHEF
Hawkins	Jon K.	Argonne National Laboratory
Hechler	Ludwig	GSI
Hendricks	Brian S.	Fermi National Accelerator Laboratory
Herb	Steve W.	DESY
Heubers	Wim P. J.	NIKHEF
Hill	Jeff O.	Los Alamos National Laboratory
Himel	Thomas M.	SLAC
Hoff	Lawrence T.	Brookhaven National Laboratory
Hofler	Alicia S.	SURA/CEBAF
Holt	James A.	Fermi National Accelerator Laboratory
Humphrey	John W.	SLAC
Ivanov	Iouri N.	Fermi National Accelerator Laboratory
Jirousek	Ivo	Paul Scherrer Institute
Johns	Ronald A.	Argonne National Laboratory
Johnson	Glenn C.	Fermi National Accelerator Laboratory
Jordan	Kevin	CEBAF
Joshel	Robert H.	Fermi National Accelerator Laboratory
Kamikubota	Norihiko	KEK
Kanaya	Norichi	ESRF
Karnaev	Serguei E.	BINP
Kasley	Paul A.	Fermi National Accelerator Laboratory
Katoh	Tadahiko	KEK
Keitel	Rolf	TRIUMF
Kelm	Louis	Kinetic Systems Corporation
Klassen	Erwin	TRIUMF
Klayum	Mark A.	Fermi National Accelerator Laboratory
Klotz	Wolf-Dieter	ESRF
Knott	Martin J.	Argonne National Laboratory
Ko	In Soo	POSTECH
Kotov	Vladislav M.	Joint Institute for Nuclear Research
Kowalkowski	Jim B.	Argonne National Laboratory
Kozlovsky	Mark M.	Fermi National Accelerator Laboratory
Kraimer	Martin R.	Argonne National Laboratory
Krakar	William J.	Fermi National Accelerator Laboratory
Krause	Udo	GSI
Krebs	Olaf	DESY
Krzywdzinski	Stanislaw M.	Fermi National Accelerator Laboratory
Kucera	Michael J.	Fermi National Accelerator Laboratory
Kuiper	Berend	CERN
Kuznetsov	Sergey	Kurchatov Institute - RRC
Lack	Michael N.	University of Richmond
Lackey	Sharon L.	Fermi National Accelerator Laboratory
Lahey	Terri E.	SLAC
Lahti	George E.	CEBAF
Laird	Robert J.	Argonne National Laboratory
Lamont	Mike J.	CERN
Lange	Ralph	BESSY II
Lasiter	Tom	Ariel Corporation
Lavender	William M.	Illinois Institute of Technology
Le Goff	Jean-Marie H.	CERN
Lécorché	Eric	GANIL

Lenkszus	Frank R.	Argonne National Laboratory
Lessner	Jeff	ICS Ltd.
Lewis	Julian H.	CERN
Lewis	Stephen A.	Lawrence Berkeley Laboratory
Li	Xingyue	Argonne National Laboratory
Liu	Chuande	Argonne National Laboratory
Lockyer	Nigel	University of Pennsylvania
Loukiantsev	Alexandre F.	Institute for High Energy Physics
Lublinsky	Boris S.	Fermi National Accelerator Laboratory
Lucas	Peter W.	Fermi National Accelerator Laboratory
Lumpkin	Alex	Argonne National Laboratory
Manwaring	William A.	Indiana University-CF
Marsh	William L.	Fermi National Accelerator Laboratory
Martz	Virginia E.	Vista Control Systems, Inc.
Masuda	Takemasa	SPRING-8
Maugain	Jean-Marie	CERN
McClure	Craig R.	Fermi National Accelerator Laboratory
McCrory	Elliott S.	Fermi National Accelerator Laboratory
McDowell	William P.	Argonne National Laboratory
McGehee	Peregrine M.	CFHT
McGinnis	Leslie	Intel/Pioneer Corporation
Meier	John C.	Creighton University
Meyer	Jens	ESRF
Mezger	Anton C.	Paul Scherrer Institute
Mikheev	Mikhail	Institute for High Energy Physics
Miller	Irvine	Heurikon Corporation
Minich	James M.	Argonne National Laboratory
Momal	Frederic	CERN
Mooney	Tim M.	Argonne National Laboratory
Mouat	Michael M.	TRIUMF
Mozin	Igor V.	STC St. Petersburg
Mueller	Klaus D.	KFA Juelich/ZEL
Müller	Roland M.	BESSY
Munson	Floyd H.	Argonne National Laboratory
Mutoh	Masakatsu	LNS Tohoku University
Myers	David R.	CERN
Nakamura	Tatsuro T.	KEK
Navratil	Jiri	Czech Technical University
Nawrocki	Gregory J.	Argonne National Laboratory
Neswold	Richard M	Fermi National Accelerator Laboratory
Nikogosian	Valeri	Yerevan Physics Institute
Ninin	Pierre F.	CERN
Olsen	Robert H.	Brookhaven National Laboratory
Oothoudt	Michael A.	Los Alamos National Laboratory
Ortega	Mario G.	SLAC
Ostiguy	Jean-Francois	Fermi National Accelerator Laboratory
Oustinov	Evgeni A.	Institute for High Energy Physics
Pasian	Fabio	Osservatorio Astronomico di Trieste
Patel	Raj	Xilinx, Inc.
Paterno	Laura A.	Fermi National Accelerator Division
Patz	Manfred	Softing GmbH
Pearson	Pauline S.	Brookhaven National Laboratory
Perriollat	Fabien	CERN
Peters	Robert E.	Fermi National Accelerator Laboratory

Plesko	Mark	Institute "Jozef Stefan"
Poole	John	CERN
Popov	Gennady F.	Kharkov State University
Pordes	Ruth	Fermi National Accelerator Laboratory
Pose	Rudolf	Joint Institute for Nuclear Research
Potepan	Franco	Sincrotrone Trieste
Prosin	Boris	Institute for High Energy Physics
Pucillo	Mauro	Osservatorio Astronomico di Trieste
Pugliese	Roberto	Sincrotrone Trieste
Rabany	Michel J.	CERN
Raffi	Gianni	European Southern Observatory
Rahn	Joachim G.F.	BESSY
Ramamoorthy	Susila	Brookhaven National Laboratory
Rarback	Harvey	The University of Chicago
Raupp	Gerhard	Max-Planck-Institut für Plasmaphysik
Rausch	Raymond	CERN
Rehlich	Kay	DESY
Reicks	Jeremy	Datacube, Inc.
Reid	David	Argonne National Laboratory
Ribeiro	Pedro M.D.	CERN
Rivers	Mark L.	The University of Chicago
Rose	Patricia A.	Los Alamos National Laboratory
Rybin	Victor	Moscow Engineering Physics Institute
Rytchenkov	Valeri	Fermi National Accelerator Laboratory
Saban	Roberto I.	CERN
Sakaki	Hironao	SPRING-8
Sathe	Smita M.	Brookhaven National Laboratory
Satogata	Todd J.	Brookhaven National Laboratory
Saunders	Claude	Argonne National Laboratory
Schaa	Volker R.W.	GSI
Schaffner	Sally K.	CEBAF
Schaller	Stuart C.	Los Alamos National Laboratory
Schloesser	Klaus	Forschungszentrum Karlsruhe
Schoessow	Paul	Argonne National Laboratory
Schütte	Winfried	DESY
Seaver	Chris L.	Argonne National Laboratory
Shea	Michael F.	Fermi National Accelerator Laboratory
Shea	Thomas J.	Brookhaven National Laboratory
Sherwin	Greg R.	SLAC
Shirakata	Masashi	KEK
Shoaee	Hamid	CEBAF
Sibley	R. Coles	Massachusetts Institute of Technology
Sicard	Claude H.	CERN
Sidorowicz	Kenneth V.	Argonne National Laboratory
Skelly	Joseph F.	Brookhaven National Laboratory
Smedinghoff	James G.	Fermi National Accelerator Laboratory
Smith	John D.	Brookhaven National Laboratory
Smolucha	John M.	Fermi National Accelerator Laboratory
Stanek	Michael W.	SLAC
Starker	Josef	Manne Siegbahn Laboratory
Stecchi	Alessandro	INFN-LNF Frascati
Stein	S. Joshua	Argonne National Laboratory
Steiner	Rudolf	GSI
Stephens	Ed T.	Fermi National Accelerator Laboratory

Stevens	Rick	Argonne National Laboratory
Stoffel	John B.	Argonne National Laboratory
Stott	John P.	University of Wisconsin-Madison
Streets	Jon	Fermi National Accelerator Laboratory
Strubin	Pierre M.	CERN
Sullivan	Joe	Argonne National Laboratory
Sutokusumo	Parangtopo	University of Indonesia
Sytin	Alexander N.	Institute for High Energy Physics
Takada	Eiichi	National Institute of Radiological Sciences
Taketani	Atsushi	SPRING-8
Tanaka	Ryotaro	SPRING-8
Tang	Johnny Y.	CEBAF
Tang	Yong-nian	Brookhaven National Laboratory
Taurel	Emmanuel	ESRF
Terekhov	Victor I.	Institute for High Energy Physics
Thomas	George	Contemporary Control Systems, Inc.
Thuot	Michael E.	Los Alamos National Laboratory
Thuresson	Leif S.	The Svedberg Laboratory
Tieman	Brian J.	Argonne National Laboratory
Ulrich	Michele	GANIL
Urban	Gregory S.	Omnibyte Corporation
Utterback	Jeff H.	Fermi National Accelerator Laboratory
Vadon	Marc	CERN
Vaguine	Alexy	Moscow Radio-Technical Institute
van Zeijts	Johannes	CEBAF
Varga	Lászl6 Z.	KFKI-MSZKI
Vincent	John J.	Michigan State University
Vineyard	Michael F.	University of Richmond
Vinogradov	Vjacheslav I.	Institute for Nuclear Research Russian Academy of Science
Voevodine	Valeri P.	Institute for High Energy Physics
Von Rüden	Wolfgang	GSI
Vong	Chi-vai F.	Argonne National Laboratory
Voyda	Gary	Allen-Bradley Company, Inc.
Waller	Alex M.	Fermi National Accelerator Laboratory
Wampler	Steve	Aura, Inc.
Wang	Junye	Fermi National Accelerator Laboratory
Watson	William A.	CEBAF
West	Robert E.	Fermi National Accelerator Laboratory
Westbrook	Mary L.	Argonne National Laboratory
White	Karen S.	CEBAF
White	Vicky A.	Fermi National Accelerator Laboratory
Whitney	Roy	CEBAF
Wieland	Susan M.	Gemini 8M Telescopes
Winans	John R.	Argonne National Laboratory
Winterowd	Lin A.	Fermi National Accelerator Laboratory
Witherspoon	Sue E.	CEBAF
Woodbury	Kerry J.	Fermi National Accelerator Laboratory
Worm	Torben	Aarhus Universitet
Wright	Michael C.	Lockheed Martin/ORNL
Wu	Hoggong	DESY
Yamamoto	Noboru	KEK
Yamamoto	Yoshitaka	Hitachi 2 Osen
Yamashita	Akihiro	SPRING-8
Yogendran	Priscilla J.	TRIUMF

Zagel	James R.	Fermi National Accelerator Laboratory
Zaitsev	Vladimir I.	Troitsk Institute for Innovation and Fusion Research
Zaroudnev	Iouri V.	BINP
Zdarko	Richard W.	SLAC
Zhao	Jijiu	Institute of High Energy Physics
Zingelman	Timothy E.	Fermi National Accelerator Laboratory

ACRONYMS AND ABBREVIATIONS

ACIS	Access Control Interlock System (APS)
ACNET	Accelerator Control NETwork (FNAL)
ADC	Analog to Digital Converter
ADO	Accelerator Device Object (RHIC)
AGS	Alternating Gradient Synchrotron (BNL)
AI	Artificial Intelligence
ALS	Advanced Light Source (LBL)
ANL	Argonne National Laboratory (Chicago)
API	Application Program Interface
APS	Advanced Photon Source (ANL)
ARP	Address Resolution Protocal
ATM	Asynchronous Transfer Mode
AUS	Accelerator Utility System (IHEP, Protvino)
BCD	Beam Coordination Diagram (CPS)
BEPC	Beijing Electron-Positron Collider
BESSY	Berliner ElektronenSpeicherring-gesellschaft fur SYnchrotronstrahlung (Berlin electron laboratory for synchrotron radiation)
BINS	Budker Institute of Nuclear Physics (Novosibirsk)
BNL	Brookhaven National Laboratory (Long Island, NY)
BPM	Beam Position Monitor
BSD	Berkeley Software Distribution
BURT	Back Up Restore Tool
CAD/CAM	Computer Aided Design/Computer Aided Manufacturing
CAL	CAN Application Layer
CAMAC	Standard crate, module and backplane system
CAN	Controller Area Network (Automobile bus system)
CAS	Central Alarm Server (CERN)
CASE	Computer Aided Software Engineering
CDEV	Common DEVice *or* Control DEVice
CDF	Collider Detector at Fermilab
CDS	Control and Data System
CEBAF	Continuous Electron Beam Accelerator Facility (Newport News, VA)
CECIL	Common EPICS CAN Interface Layer
CERN	European Organisation for Nuclear Research (ex-Centre European pour le Research Nucleaire)
CICERO	Control Information system Concepts based on Encapsulated Real-time Objects (CERN)
CIM	Computer Integrated Manufacture
CLC	Console-Level Computers (RHIC)
CORBA	Common Object Request Broker Architecture
COSY	Cooler Synchrotron (Juelich, Germany)
CPS	CERN PS
DAC	Digital to Analog Converter
DANTE	DAphne New Tools Environmant (Frascati)
DAQ	Data AcQuisition board (CERN)
DAQ	Data AcQuisition System (LBL)
DBMS	DataBase Management System
DCE	Distributed Computing Environment
DCM	Direct Communication Module (Allen Bradley)
DDS	Direct Digital Synthesis

DEC	Digital Equipment Corporation
DESY	Deutches Electronen SYnchrotron (Hamburg)
DIC	Device Interface Computer (PAL)
DIO	Digital Input/Output
DSC	Device Stub Controller
DSP	Digital Signal Processor
DTM	Distributed Table Manager (CERN)
ECA	Equipment Control Assembly (CERN)
ECR	Electron Cyclotron Resonance (Ion Source)
EIU	Equipment Interface Unit (ELETTRA)
EPA	Electron-Positron Accumulator (CERN)
EPAC	European Particle Accelerator Conference
EPCS	Experimental Physics Control Systems (Interdivisional Group of EPS)
EPICS	Experimental Physics and Industrial Control System
EPLD	Electrically Programmable Logic Devices
EPS	European Physical Society
ESO	European Space Organisation *or* European Southern Observatory
ESOC	European Space Operations Centre
ESRF	European Synchrotron Radiation Facility
EU	European Union
FDDI	Fiber Distributed Data Interface
FEC	Front End Computer
FNAL	Fermi National Accelerator Laboratory (Fermilab)
FPGA	Field Programmable Gate Arrays
FSM	Finite State Machine
GANIL	Grand Accelerateur National d'Ions Lourdes (Heavy Ion Accelerator, Caen, France)
GPIB	General Purpose Interface Bus (IEEE-488)
GSI	Gesellschaft fuer SchwerIonenforschung (Heavy Ion Research Laboratory, Darmstadt, Germany)
GUI	Graphical User Interface
HCI	Human Computer Interface (CERN)
HEP	High Energy Physics
HERA	Hadron-Elektron Ring Anlage (proton and electron colliding rings, DESY)
HIRFL	Heavy Ion Research Facility in Lanzou (China)
HMI	Hahn-Meitner-Institut (Berlin)
HVAC	High Voltage Alternating Current *or* Heating, Ventilation and Air Conditioning
ICALEPCS	International Conference on Accelerator and Large Experimental Physics Control Systems
ICMP	Internet Control Message Protocol
ICRR	Institute for Cosmic Ray Research (Japan)
ID	Insertion Device
IDC	Intelligent Device Controller (BINS)
IDT	Interface Development Tool
IHEP	Institute for High Energy Physics (Protvino Russia *and* Beijing, China)
ILC	Intelligent Local Cotnroller
ILL	Institut Laue Langevin (Grenoble)
I/O or IO	Input/Output
IOC	Input/Output Controller
IP	Internet Protocol *or* Industry Pack

IPC	InterProcess Communication
ISAS	Institute of Space and Astronautical Science (Japan)
ISN	Institut des Sciences Nucleaires (Grenoble)
ISO	International Standards Organization
ISR	Interrupt Service Routine
ISS	Instrumentation SubSystem (PSR)
KEK	National Laboratory for High Energy Physics (Japan)
KFA	KernForschungsAnlage (Nuclear Research Laboratory, Juelich, Germany)
KEKB	KEK B-Factory
KFTI	Kharkiv Institute of Physics and Technology (Ukraine)
LAN	Local Area Network
LAMPF	Los Alamos Meson Physics Facility
LANL	Los Alamos National Laboratory
LANSCE	Los Alamos Neutron Science CEntre
LBL	Lawrence Berkeley Laboratory (Berkeley, CA)
LCR	Local Control Room
LEAR	Low Energy Antiproton Ring (CERN)
LEP	Large Electron-Positron collider (CERN)
LHC	Large Hadron Collider (proposed, CERN)
LIL	LEP Injector Linac (CERN)
LINAC	LINear ACcelerator
LPC	Local Process Controller (ELETTRA)
MAD	Methodical Accelerator Design program
MAP	Manufacturing Automation Program
MBS	Message Broadcasting System
MCR	Main Control Room,
MEDM	Motif Editor/Display Manager
MEPHI	Moscow state Engineering PHysics Institute
Mil-1553	MIL/STD-1553B serial field bus
MMI	Man-Machine Interface
MPX	MultiPleX system (CERN)
MRTI	Moscow RadioTechnical Institute
MTG	Master Timing Generator
NAO	National Astronomical Observatory (Japan)
NC	Network Compiler (CERN) *or* Numerically Controlled
NFS	Network File System
NIKHEF	National Institute for Nuclear and High-Energy Physics (Amsterdam, Netherlands)
NIRS	National Institute of Radiological Sciences (Chiba, Japan)
NODAL	NOrsk Data Accelerator Language (CERN)
NSLS	National Synchrotron Light Source (BNL)
OAT	Osservatorio Astronomico di Trieste (Italy)
ODBC	Open DataBase Connectivity (KEK)
ODBMS	Object DataBase Management System
ODS	Open Database Services (KEK)
OLE	Object Linking & Embedding
OO	Object Oriented
OOD	Object Oriented Design
OODB	Object Oriented DataBase
OOP	Object Oriented Paradigm
OOT	Object Oriented Technology
OPI	OPerator Interface
OS	Operating System

OSD	Open Database Services
OSF	Open Systems Foundation
OSI	Open Interconnection Standard
PAC	Particle Accelerator Conference
PAL	Pohang Accelerator Laboratory (Korea) *or* Programmable Array Logic
PC	Personal Computer (usually IBM or clone)
PCA	Process Control Assembly (CERN)
PHIGS	Programmer's Hierarchical Interactive Graphics System
PID	Proportional, Integral and Differential Controller
PLC	Programmable Logic Controller
PS	Proton Synchrotron
PSB	PS Booster (CERN)
PSI	Paul Scherrer Institut (Villigen, CH)
PSR	Proton Storage Ring (LAMPF)
PSS	Personnel Safety System (APS)
PV	Process Variable
RDBMS	Relational DataBase Management System
REMOT	Remote Experiment MOnitoring and conTrol (EU Project)
RHIC	Relativistic Heavy Ion Accelerator (BNL)
RISC	Reduced Instruction Set Computing
RMON	Remote MONitoring protocol
RPC	Remote Process Call
RTP	Real-Time Protocol (Internet)
SA/SD	Structured Analysis/Structured Design
SCC	Spring-8 Control Command
SCI	Simple CAN Interface
SCMP	Software Configuration Management Plan
SDDS	Self Describing Data Sets (ANL)
SDFD	Self Describing File Protocol (ANL)
SFC	Sector-Focussing Cyclotron
SLAC	Stanford Linear ACcelerator (Stanford, CA)
SLC	SLAC Linear Collider
SNL	State Notation Language
SNMP	Simple Network Management Protocol
SOSH	SOftware SHaring
SPring-8	Super Photon ring 8GeV
SPS	Super Proton Synchrotron (CERN)
SQA	Software Quality Assurance
SQL	Structured Query Language
SR	Synchrotron Rariation
SRRC	Synchrotron Radiation Research Center (Taiwan)
SSC	Separate-Sector Cyclotron
SSRL	Stanford Synchrotron Radiation Laboratory
STAR	Solenoidal Tracks at RHIC
SVVP	Software Verification and Validation Plan
TCP/IP	Transport Control Protocol/Internet Protocol
TCR	Technical Control Room (CERN)
TDM	Time Division Multiplex
TESLA	TEv Superconducting Linear Accelerator (DESY and others)
TNG	Telescopio Nazionale Galileo (La Palma)
TOD	Time of Day
TPC	Time Projection Chamber (RHIC)
TRINITI	TRoitsk INstitute for Innovation and Thermonuclear

	Investigations (Russia)
TRISTAN	Transposable Ring Intersecting STorage Accelerator in Nippon (KEK electron-positron collider)
TRIUMF	TRI-University Meson facility (Vancouver, British Columbia, Canada)
TSS	Telescope Software System (TNG)
UCS	Utility Control System (IHEP, Protvino)
UDP	User Datagram Protocol
UIDS	User Interface Development System
UIMS	User Interface Management System
UMMI	Uniform Man Machine Interface
VEPP	Electron-Positron Collider (Novosibirsk, Russia)
VLAN	Virtual Local Area Network
VLT	Very Large Telescope (ESO)
VME	Versa Module European (Crate, module & backplane)
VMEbus	Often used for VME. Not just the VME backplanes.
VMS	Virtual Memory System (DEC)
VUV	Vacuum Ultra-Violet
VXI	VME EXtension for Instrumentation
WAN	Wide Area Network
WIMP	Windows, Icons, Menus and Pointer Interface
WYSIWYG	What You See Is What You Get
WSS	Workstation Software System (TNG)
WWW	World-Wide Web
XUIMS	X-Windows User Interface Management System (CERN)

FERMILAB-Conf-96/069

Table of Contents

* Invited Papers

* Invited Papers

* Invited Papers

* Invited Papers

* Invited Papers

HOW CAN OPERATIONS GET THE APPLICATIONS SOFTWARE THAT THEY WANT ?

R. Bailey, CERN, CH-1211 Geneva 23, Switzerland

In the SL division at CERN, any major development of application software to be used in the control room entails interaction and a division of responsibilities between people from operations, controls, accelerator physics and equipment groups. Providing the suite of applications for the operators to use for efficient exploitation of the machines is no exception to this. Indeed, since routine operation accounts for the largest fraction of machine time, it should be regarded as the most important activity. We have found that teams led by operations people, making pragmatic use of formal development methods, can be very effective in producing the software required for running the machines. This paper examines the modus operandi *of some operations-driven projects, and contrasts this with some less successful ventures.*

1 INTRODUCTION

The successful production of satisfactory application software for driving accelerators has remained a difficult goal for years. This is true over a wide range of laboratories and over the different categories of user. Why is this so ? In an attempt to throw some light on this perennial problem, the question is broken down into three;

- who should be considered the main user of a control system ?
- what features are required ?
- how can the desired functionality be achieved ?

2 WHO SHOULD DEFINE A CONTROL SYSTEM ?

Any accelerator control system is used at different times by different personnel. During the latter stages of the construction phase, and later on for detailed investigations of faults, many equipment specialists need to run extensive tests of their hardware. During commissioning, and later on in machine development, accelerator physicists need a wide variety of utilities to measure and manipulate the beam. Once the machine is up and running, operations need a comprehensive suite of applications software for the long-term exploitation of the machine. These different groups of people each have very different and often conflicting requirements and, while a good control system should try to satisfy all kinds of user this is rarely, if ever, the case.

Where should the priority be? Software for equipment testing is needed before software for beam commissioning, which in turn is needed before software for routine operations, and this chronology determines the order in which applications have to become available. However, while the priority in terms of the calendar is clear, the priority in terms of machine time is not. The equipment testing and beam commissioning phases are hopefully rather short, amounting to something like six months each. Operational running of the machine, on the other hand, is hopefully rather long, with ten years a typical minimum. Within the years of routine operation there are equipment faults to be diagnosed and machine development periods to be exploited, but again these should constitute a small fraction of the whole. In a typical year of LEP operation, for example, the machine time is divided into 70% on routine operation, 20% on machine development and 10% on fixing problems.

Seen in this light (and there are others [1]), the requirements of the operators should be of prime importance. In fact, routine operation is the reason the machine was constructed in the first place, with commissioning a necessary step along the way. Unfortunately, when planning what will be needed to run a machine, most application development projects concentrate on the earlier but shorter phases, fading out once the machine is successfully commissioned and neglecting completely the longer term. A new initiative, taking considerable time and requiring new resources, is then needed before operations are able to run the machine effectively and efficiently over many years.

3 WHAT FEATURES DO OPERATIONS WANT ? - THE BASICS

Leaving aside for the moment the crucial question of the detailed functionality required to run the machine, two things set operations personnel apart from other users of the control system. Firstly, the scope of their work necessarily covers the whole machine rather than specialist areas, and in this they prefer a high level of standardisation across different applications and even, in some cases, across different accelerators. Secondly, machine operation happens 24 hours a day, seven days a week, typically for 30 to 40 weeks each year and the 'comfort' provided by the control system is extremely important.

A. Standardisation

Having a suite of applications that conform to well defined standards can contribute enormously to an operator's efficiency. The basic interaction with different applications should be required to follow a set of rules specified by operations. This would enable all the essential functions (start, stop, pop, push, reduce, enlarge, ...) and some slightly deeper functionality (display handling, parameter input, task and sub-task selection, ...) always to be performed in the same way. With this, navigation around the system and a large part of the operator's interaction with it would be standard. One might say that this is obvious, but a lot of software presently used to control the SPS, for example, would fail to meet these basic requirements. Why is this so ? The reason for this shortcoming is nearly always due to the fact that the software in question was produced either by, or following specifications from, equipment specialists, and was then made available to operations. Such software may be fine for the individual needs of the equipment specialist, but it does not fulfill the more global requirements for operations.

B. Ease of use of interactive applications

Listed here are a few of the characteristics of user-friendly applications software:

- Reliability
- Stability against change
- Error reporting
- On-line help
- Execution speed
- Data on a need-to-know basis

Again one might think that these requirements are obvious, but at three o'clock in the morning they often seem to take on an extra significance.

C. Fixed displays

Operations is not all interactive. During stable conditions an accelerator often runs without any operator intervention. What the operator needs at these times is a good fixed display system, giving him at-a-glance access to a rather large amount of data which tell him either that all is running well or that something has gone wrong. Display of information of this kind on a permanent basis, with a regular update, contributes greatly to the operator's comfort.

Much use is made of these facilities in the control room from where both the SPS and LEP are controlled. For stable LEP operation there are about 15 fixed display screens, many of them divided into 4 or sometimes more windows, and many of them configurable to suit the operator's needs. For the SPS the number is a little smaller but the system is still regarded as essential for operation of the machine.

D. Alarms

A particular example of a fixed display is the alarm screen, which is again of great importance for both SPS and LEP operation. There are two alarm displays for each machine, and they are used in two modes. During stable operation, the alarm screen is the first place to look if something abnormal happens before searching interactively. During a startup, on the other hand, the screen is used as a kind of check list; removal of alarms from the screen implies getting closer to a working machine. In both cases, the integrity of the system is of great importance. An alarm system in which one does not have 100% trust is almost useless. First for SPS and later for LEP, it took years of operational experience to get a viable system which the operators could use with confidence.

4 HOW CAN OPERATIONS GET THE FUNCTIONALITY THAT THEY WANT ?

An accelerator is a collection of many different systems, and a common approach is to provide applications software on a system by system basis. Working to agreed standards as outlined in the previous section, it should be reasonably straightforward for operations to specify what is needed sufficiently well for personnel outside operations to produce these applications. Many accelerator control systems are produced in this way, in some cases with a high degree of success [2]. Of course, all the underlying facilities of networks, communication protocols and so on have to be present, but this is taken for granted here.

However, if the operations team wants a coherent and integrated suite of applications, specification of the required functionality becomes much harder and can only be achieved if the whole system is conceived as a whole. This is an important step to take, and it is in this domain where the SL division at CERN has had good experience of teams involving operations people at all stages of the development process, first for the SPS and later for LEP [3]. A brief history of this experience is described in some detail for the SPS, with additional remarks about the LEP software development where differences arose.

I should point out that in SL division, operations means people who work regularly on the machines but who do have time available, particularly during long machine shutdowns, for other things such as software development. Furthermore some of the operations personnel are engineers or physicists with several years of hands-on experience of running the machine(s).

A. SPS

The SPS, commissioned in 1976, ran purely as a proton machine for a fixed target physics program until the early 80s. Then followed several years where about half of each year was devoted to proton-antiproton collider operation, mostly at a stable beam energy, but also in 1985 as a pulsed collider allowing higher energy collisions to be achieved. This evolution meant many modifications to the application programs, which underwent considerable and somewhat piecemeal development to accommodate the different modes of operation. This resulted in software that did the job, but that became difficult to maintain and almost impossible to modify. Consequently in 1985, when considerations began for further important changes to allow SPS to be used as the injector for LEP, it soon became clear that a major software rewrite was inevitable.

A small team of operations group physicists with a history of software development were mandated to look into the problem. The overall aim of the project was to provide, for early 1988, a software system for the operation of SPS with cycles of electron and positron bunches for LEP interleaved with proton or ion acceleration cycles for the ongoing SPS physics program. This would involve a move to a completely new platform on which to run the new software. The mandate also called for the establishment of clear guidelines for production of any new applications software for SPS and at a later date for LEP. With such a major and wide-ranging project, it was decided to look outside the accelerator sector for ideas. Through the seventies and early eighties the software industry had seen the increasing imposition of discipline on development activities, resulting in turn in the techniques of structured programming, structured design and structured analysis. These techniques had been introduced not on theoretical grounds, but were based on the desirable aspects found in a wide range of large, successful projects. Furthermore there were already groups at CERN who were beginning to use these techniques, and it was decided to follow the recommendations coming from these sources [4].

The team began by performing a detailed structured analysis of the various applications existing at the time to drive the SPS, using dataflow diagrams to describe what the software did. This allowed a better understanding of the details of the running system and brought out clear areas that were in common to all accelerator systems. Requirements for the new system were then collected and used together with the analysis of the old to produce a model of the desired software. This model, again expressed in terms of dataflow diagrams and now with a full supporting data dictionary, represented a detailed description of the functionality that the new software should have, and formed the cornerstone of the project from then on. It was not easy to produce, taking more than a year for a core team of three people with significant contributions from many more. It must be said, however, that part of this time was spent getting organised and developing expertise in a new field.

It was realised that one can only go so far in way of developing the functionality of the system before the structure of the underlying data has to be understood. Data modeling therefore was a distinct phase to be completed before any further progress could be made. This resulted in a fully normalised description of the data needed to run the SPS in multicycling mode, and largely defined the design of the on-line database to be implemented to drive the software.

Armed with a comprehensive model of the required functionality and underlying data, design could start. Since the problem was well understood and thoroughly described, it was straightforward to divide the system into several parts. An important aspect of the way this partition was made was that personnel from the analysis team undertook the responsibility for the design and eventual coding of the different subsystems. The team all agreed to continue following a rather formal approach, using structured design techniques to develop structure charts of the various parts of the system before producing any software modules. This meant that the project, running since mid 1985, had still to deliver a single line of code by late 1987, a detail that was not lost on the project management at the time !

The importance of having the same people involved throughout cannot be overstressed. It avoided potentially disastrous pitfalls in the handover between analysis and design, and it instilled in each individual a huge sense of purpose that the whole product, from ideas through to implementation, should be delivered on time. In this respect the motivation of the different members of the team was very high during the difficult period of implementation according to schedule.

Implementation was made incrementally, with the core system in place for the LEP octant injection test of summer 1988. Definition of the operator interface was among the last and quickest things to be done, and was achieved using initial specifications from operations, rapid prototyping and iterative development. The result was, and still is, an operator interface very much appreciated by all users [5]. All major parts of the system were in place for LEP commissioning in mid 1989, with more functionality added over the following years.

After several years, the suite of software discussed continues to form the heart of SPS applications and is still generally regarded by operations as excellent. Nearly all personnel involved in the original development are no longer at CERN, and maintenance and further development is now carried out from the controls group.

B. LEP

After the first year of running LEP, it was clear to several people in operations that the application software used for commissioning would need considerable improvement for efficient long-term exploitation of the machine [6]. Following the example of the SPS project, an initiative was launched in 1990 to attack the problem. A small team was assembled, again with a nucleus of three operations group physicists having a history of software development but in this case with early involvement of three controls personnel. Functional decomposition of the problem was performed along similar lines to the SPS analysis, until once again it was clear that the data structure needed to be clearly defined. In this case, entity relationship modeling techniques were used to organise the data, leading to a database design that was well suited to a commercial relational database system and the choice of ORACLE for implementation [7].

In a similar way to the SPS, the system was then divided between several people on the team, but in this case the development from there on was not required to follow any (semi-)formal methods. Some people chose to use structured design to develop their ideas, others did not. More important was again the clear definition of responsibility for the different parts of the system, and the motivation and determination to deliver on time.

Initial implementation of the LEP software for the 1992 startup was more sweeping, reflecting a more complete suite of applications software, than in the case of the SPS. The operator interface drew heavily on both the ideas and the existing packages from the SPS, with all new developments made to the same standards, resulting in a highly uniform product across two very different machines. Developments since 1992 have mostly been to accommodate different modes of operation of the machine involving either new accelerator systems or existing ones driven in a new way. The main exception was the introduction in 1994 of a comprehensive beam measurement and fixed display system, now considered essential for routine operation.

C. Some comments

The two development projects described had somewhat different origins. The first was a necessity in that something major had to be done in order to use the SPS as LEP injector, while the second arose largely out of operations' discontent over the tools that were (not) available to do their job. However, in both cases it was felt that an approach covering the whole accelerator was needed to produce applications software that was more coherent than in the past. Having decided this, it was clear that operations were best placed to define what the software should do. One may question whether it is necessary to be so ambitious, but in both cases the approach turned out to be justified in that it has resulted in an integrated suite of data-driven application software. As well as the undeniable advantage of having such an integrated system, there have been additional benefits such as that on both machines a new accelerator system can often be introduced without writing any code at all.

It is difficult to see how such accelerator-wide facilities could be specified without significant involvement of the people who drive the machine, but should their involvement stop there ? Ideally for operations the answer is yes, but in the two cases discussed the output of the analysis phase, while comprehensive, was very difficult to understand. This was particularly true in the SPS case, which perhaps represents a failing since one of the claims of structured decomposition is to control the complexity of a large system. In any case, to hand a specification of this kind over to a design team to develop and implement would introduce considerable overheads in the development process. Whether done formally or not, there are several iterations required around the analysis/design loop. In fact, this is the most critical phase of any project whose duration runs to several man-years. If the specifications and the implications thereof are not fully understood by the design and implementation team, the delivered product will never be up to standard. Having the same people involved throughout will at the very least allow important savings at this stage, and at most could prove the salvation of the project.

An illustration of this particular problem is to be found in the original software development for LEP, where one group of people produced specifications and handed them over to a second group for design and implementation. While the specifications contained all the ideas of the commissioning team, something was lost in the handover and a design of the *specified* software was never done. This resulted in a minimal system used to commission the accelerator, all of which was subsequently rewritten, as described above.

D. Use of formal methods

The formal methods adopted in the SPS and LEP projects were always seen as a means of getting the job done, not because of any desire to impose a particular way of working. In fact in both projects, once the software was in place the analysis and design documents have not been kept up to date with the code. This means that these documents no longer describe what the software does and how it does it, and maintenance is still a potential problem if personnel leave.

Having said this, it is difficult to see how either of the two projects described could have succeeded without the use of some formal technique or other. Structured analysis and design may not be the best method; it certainly has several weaknesses and to follow it rigorously could cause a project to lose momentum and stagnate.

A method, like any other tool, has to be understood and adapted to suit the needs of the situation. There is sometimes a tendency to insist on strictly following industry standards when it is not necessary to do so, and this can do more harm than good as the following example shows. An equipment specialist produced a rather simple specification for a piece of software to be developed by an application programmer. After some months this had been developed into a heavy formal document that was handed back to the equipment specialist for approval before development could begin in earnest. This required a significant investment by the end user that he was simply not prepared to give since he did not see the point. This project faltered to a halt because of this and had to be restarted from the original simple specification.

5 CONCLUSIONS

The operations group should be regarded as the most important client of an accelerator control system simply because over the lifetime of the facility they are by far the main users of the system. However, at the start of routine operations the applications available are usually inherited from equipment and machine specialists who used them during the construction and commissioning phases. These applications are generally not sufficient for operations and a new software project is then needed. Whether or not this can be avoided is not clear. It may be very difficult to say in advance what is needed for routine operation, although there is plenty of experience available from running machines for guidance. What *can* be done is to recognize the existence of the problem and ensure that sufficient resources are available, be it sooner or later, to provide the required software.

There are many basic features that any application software should have which would make the operators' work much easier and therefore more efficient. Many of these features are often overlooked during the software development process.

Small development projects, say those measured in a few man-months, should be rather easy to handle. There is no need to consider formal methods if the user and the developer can agree on a specification by other means. A clear set of guidelines and a simple specification should in many cases suffice. In fact, introducing formal methods in such a case may achieve nothing more than slowing the development down, even to a standstill.

For large projects, those measured in man-years, some kind of formal technique should certainly be considered. However, if you *are* planning to use formal techniques you had better learn how to use them. Prototype something from beginning to end before embarking on a big project. It is very easy to get stuck and produce nothing. Never try to impose formal techniques on unwilling members of the team.

It is important for the project management to realise that there are certain distinct phases in any large development. Functional decomposition and data modeling are best done on a project-wide basis, and should be essentially complete before partitioning is considered. Subsequent design, coding and implementation techniques are best left to the people or teams concerned, unless the eventual maintenance of the software is considered a serious problem. If for any reason this is an area of concern, it may well be necessary to require documentation in some serious form, which would strengthen the case for using formal techniques further into the project.

For projects covering many aspects of the accelerator, the software development team must include operations people at all stages of the development process. They are certainly needed to produce the detailed specification of the software, and continued involvement allows many corners to be cut in the later phases of the development. In both of the projects discussed in this paper operations people produced most of the code, which is not ideal for many reasons, but bypassed the problem of handing off specifications or designs to others. A better model of how to staff such a development would see a sharp decline in operations' involvement through the design phase, with a corresponding increase in effort from software engineers from controls (see Figure 1). This would open up the way for proper management of the finished code, allowing version control and maintenance in general to be handled fully by controls.

Operations people are probably best placed to perform the overall coordination of large application software projects, because they have an accelerator-wide view of the problems, because they know what they want and because they have a vested interest in getting the end product to use. Alternatively this role could be played by a software engineer from the controls group, but to be effective he or she would have to develop a sound understanding of what controlling an accelerator is all about.

Finally a word of warning; operations cannot and should not try to do everything. In many laboratories the operations staff have little or no time for software development, and even in the best cases their availability is limited and sporadic. Significant and sustained support is needed by full time software professionals. With the right balance and the right spirit of collaboration, impressive results can be achieved. Finding this balance continues to be the elusive goal for the manager of any large software project in accelerator control.

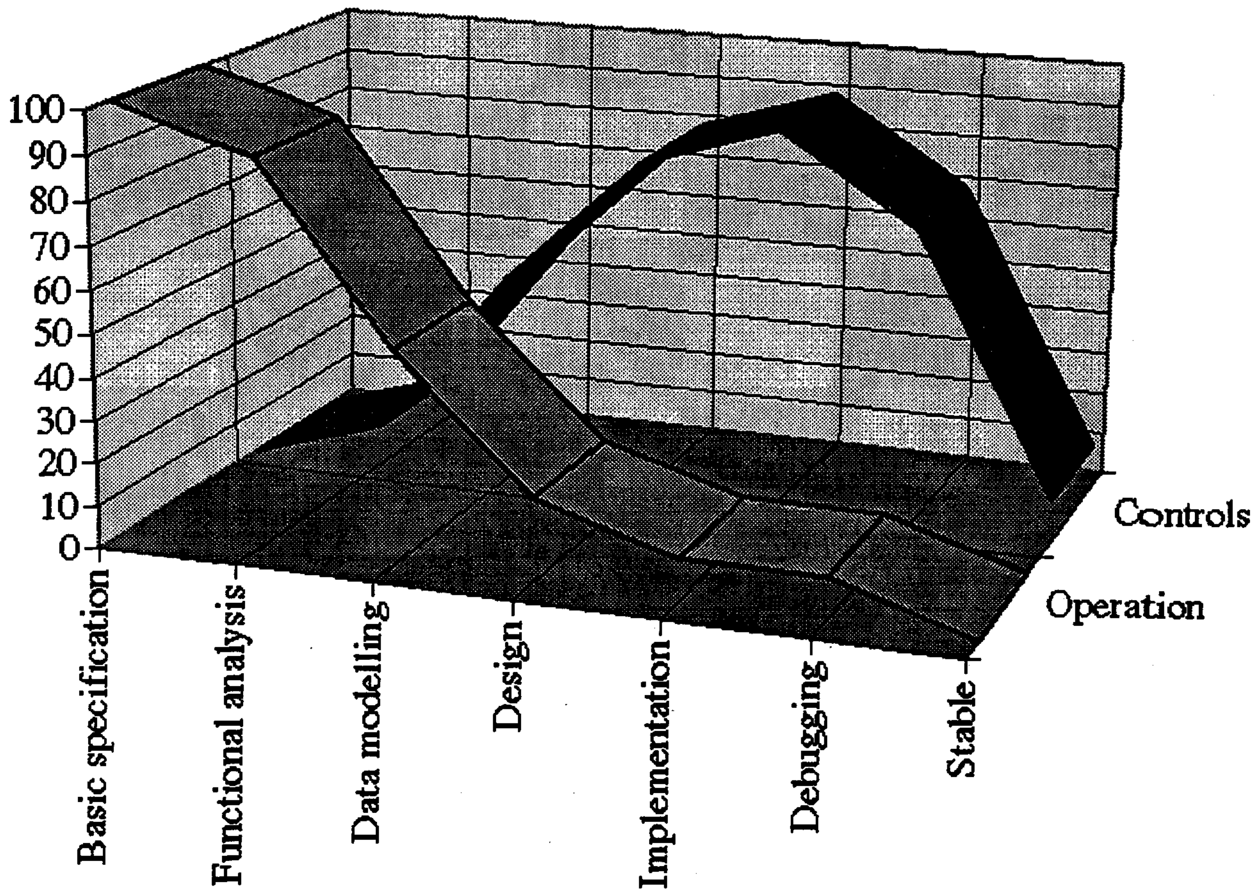


Figure 1: Ideal evolution of operational and controls involvement through the various phases of a large applications software project

6. REFERENCES

[1] E. McCrory, these proceedings
[2] M. Stanek, these proceedings
[3] M. Lamont, these proceedings
[4] I.T. Wilkie et al, Proc. ICALEPCS, Villars 1987
[5] A. Ogle et al, Proc. ICALEPCS, Vancouver 1989
[6] R. Bailey, Proc. ICALEPCS, Tsukuba 1991
[7] R. Bailey et al, ICALEPCS, Berlin 1993

Who Gets to Specify the Control System?*

by Elliott McCrory†, Jean-Francois Ostiguy and Shekhar Mishra
Fermi National Accelerator Laboratory
Batavia, Illinois, 60510, USA

1. Introduction

The users of an accelerator control system are typically considered to be: 1. the operations experts, 2. the machine engineers, 3. the accelerator scientists and 4. the administration of the facility. Each of these groups places different demands on a control system, each (except possibly the administrators) putting an equivalent load on the overall system. The operator needs a reliable and simple system every minute, but the demand at any time is not very high. The engineer and the scientist may need a lot of information, but only at very specific times, depending on the nature of the situation. This paper considers the demands on the control system from each of these groups, and how the needs of one group may tend to override the needs of another.

Firstly, we discuss the historical context in which control systems have developed, pointing out where things have changed significantly. Secondly, the four groups of workers which can potentially place requirements on the control system design are defined and their demands are considered. The interplay of the sometimes-conflicting demands of these groups is discussed. Thirdly, using Fermilab's recently completed 22-month run as an example, the amount of time spent in various aspects of a run is derived. Finally, some conclusions are drawn and recommendations are made.

A preliminary definition of terms is necessary. A "run" is an extended operating period of the complex in which no major changes are made to the complex itself. At Fermilab, a run may last for many months. The "complex" refers to the collection of all the components in the overall facility. A "component" includes everything which makes up the complex, from entire accelerators (the Tevatron) down to the small pieces of apparatus (a trim magnet). The "Controls Group" is the administrative organization whose primary function is to design, build, commission, maintain and improve the hardware and software in the complex which allows a worker (or an automated computer system) to view the operation of the complex from a location which is possibly removed from the place at which the observables are measured, for example, the Control Room.

2. Historical Perspective

The accelerator control systems of the 1970's evolved out of the work of the individual machine groups in the Fermilab complex. The Controls Group was formed in the 1980's in an attempt to integrate all of these various and independent control systems together into an "Accelerator Control System." The primary goal in this effort was to unify the control system in view of the disparate machines. This was an in-house effort, with no external commercial software, hardware or expertise.

Throughout this period, operating the complex was the overriding requirement for the control systems. It was necessary to derive paradigms for operations so that the operator, engineer and scientist did not have to learn a different paradigm for each accelerator or for each component. This effort has been extremely successful.

In the 1990's, the situation has changed substantially. In particular, the Accelerator Controls Groups are no longer on the leading edge of developing computer control technologies. This distinction has moved to Industry so that the newest technologies are no longer being developed for science.

* *Work supported by the US Department of Energy under contract number DE-AC02-76H03000*
† *Presenter*

It is a fact that over the last twenty years, computer technologies (CPUs, networks, software tools, etc.) have doubled in performance every three years. There is every indication that this trend will continue. Nevertheless, accelerator control systems remain as good examples of excellently-integrated and stable computer systems.

Our facilities, and consequently our control systems, are under great pressure to improve even though our resources are shrinking. In this light, it is reasonable to take a step back and rethink what demands are being put on our complexes and, possibly, restructure our efforts to match.

3. Who are Potentially the Specifiers?

Who are those who can potentially make specifications for the control system? In this paper, we define those who can make these specifications as: the operations experts, the machine engineers, the accelerator scientists and the administrators. These groups represent a partially orthogonal set of concerns which spans the space of possible requirements. An individual worker can be in more than one of these classifications, and, in principle, can be in all four. The following sections define these types of workers in the environment of an accelerator complex. The definitions are based on how people are organized at Fermilab.

3.1 Operations Experts

These workers are responsible for keeping the complex running 24 hours per day, for a large part of year. They are typically on rotating shifts and are usually the youngest and most enthusiastic workers at the facility. They are trained to do routine monitoring and tuning of the complex and its components. They also can repair a large class of problems which arise, but typically, the solutions to these problems are well-defined, like how to change a faulty piece of equipment. Often, it is not possible for the operations expert to fix a problem, so he/she must know when to call an expert.

The operations expert needs to have a broad view of the entire complex, but depth of knowledge of any individual component is not necessary. Also, his/her temporal view is limited: How can this problem be solved today (before my shift is over)? Can we do something so this problem does no recur?

The operations expert demands a stable and consistent view of the complex through the control system. He/she needs application programs which inform him/her of problems ("Comfort Displays"), which automate tedious, repetitive and/or intricate operations ("Sequencer"), and which make all data available at any time ("Parameter Page").

A partial list of the types of data which the operations expert requires includes: Beam currents, particle intensities, ground currents, luminosities, beam emittance and component status.

3.2 Machine Engineers

Machine engineers are the technicians, engineers and scientists who build, maintain and fix specific components in the complex. They usually are not on shift, but may be on call 24-hours per day. In an accelerator, these components include sub-accelerators (like the Main Ring or Booster), the magnet systems (the magnets, the power supplies and the connections between them), the RF systems, the cryogenics systems, the beam diagnostics and the vacuum systems, to name a few. These workers are required to fix their equipment when it is broken, write embedded-system software, write some component-specific application programs, implement improvements in the components and assist in the design of new components.

The machine expert needs to have a clear, uninterrupted view of his/her component. His/her attention span is shorter and longer than that of operations: fixing a piece of equipment takes an hour, but maintaining it takes months or years. The time span for improvements is longer still.

The machine expert demands to have total access to the internals of his/her component, often when standing in front of that component, but also from his/her office or the Control Room. He/she also needs high flexibility during the implementation phase of his/her component. Rapid prototyping of their equipment through a graphical display is desired.

The machine expert needs the same type of data as the operations expert, but his/her view tends to be more concerned with trends in these parameters.

3.3 Accelerator Scientists

Accelerator scientists are the engineers and scientists who integrate new features into the complex and who are required to diagnose and subsequently fix the really hard problems which occur during the course of the run. They are also required to understand and verify the theoretical limits of the complex, to specify future direction for the complex, to suggest software and hardware improvements and to design new components. To do these things, accelerator scientists are often required to perform experiments on the complex or on specific components in the complex.

An accelerator scientist needs to have a clear view of the entire complex, as does the operations expert, and, from time-to-time, must also have the myopic perspective of the machine expert. He/she has an attention span of the operations expert ("fix it today and for the run") and, additionally, somewhat longer, too ("how can we fundamentally improve *this* in the long term?")

The accelerator scientist demands time-correlated data. In fact, he/she demands that the control system behaves like a data acquisition system. But, primarily, he/she must have an extreme level of flexibility in controls, adding new, possibly temporary hardware and software to the system in order to diagnose a specific problem. He/she needs to be able to get data for a different type of computer in order to solve unusual problems (e.g., putting a PC into a UNIX-oriented console environment).

The type of information needed by the accelerator scientist includes: emittance growth, luminosity lifetime, integrated luminosity and general performance limitation.

3.4 Administrators of the Complex

This group of workers is included for completeness, since the demands which they place on the control system are small. In this definition, the administrators of the complex listen to the other three groups of workers and, based on the data presented from these groups (data which is usually obtained through the control system), they decide the future direction of the facility. Their need for a continuous, reliable overview of the status of the complex is obtained, often through great effort, through a semi-static, site-wide display. At Fermilab, this is realized through a closed-circuit TV channel 13.

4. Interplay of Demands

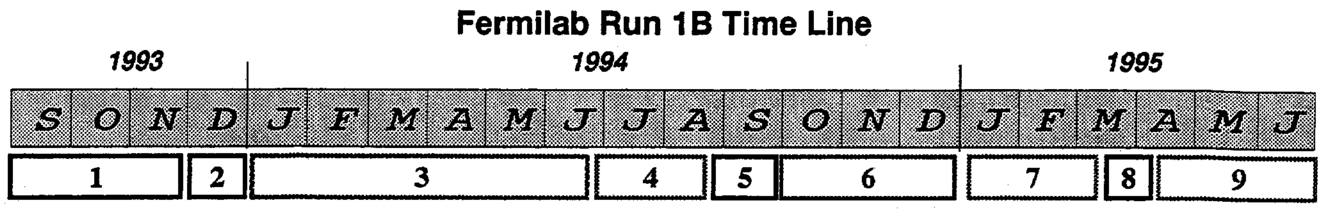
To summarize, the operations expert demands a stable, consistent view of the complex so that problems in the complex can be easily identified. The machine expert demands to have a clear view of his/her piece of equipment and needs a lot of flexibility to choose the latest and most modern tools within that equipment. The accelerator scientist demands flexibility, especially during commissioning, and good data acquisition functionality.

The key area in which these demands collide is the operations experts demand for consistency and the machine experts and the accelerator scientists demand for flexibility. In the past, this conflict has been "resolved" by saying that any of the demands of the machine experts and the accelerator scientists must be implemented in a way that does not violate the operations experts demand for consistency. This has all-too-often meant that the specific needs of the machine expert and the accelerator scientist have not been met because a canonical solution cannot be found. In particular, there has been a reticence to allow many commercial software packages to be included in the control system because they are typically difficult to integrate fully into the existing system. This makes rapid prototyping and "temporary" software for experiments rather difficult to implement. For example, if an machine expert is having difficulty finishing the installation of his piece of equipment, he/she needs to be able to use whatever tools are at his/her disposal to solve the problems. If these tools include software which is not fully integrated into the control system, then it may be difficult to use the software, and, at the worst, he/she cannot get the needed information at all.

5. How is Time Spent During a Run?

What fraction of the time is *really* spent in operations? In order to get an accurate breakdown of how time might be spent during a run, we call upon our own direct and recent experience with the just-completed run of the Fermilab Collider "Run 1B." It began in September of 1993 with the commissioning of the 400 MeV Linac and ended in June, 1995:a total of 22 months. The goal of this run was to increase the integrated luminosity delivered to the experiments each month by about a factor of two over the best performance of the previous run. Moreover, we were scheduled to operate the complex for about twice as long as in the previous run, so the experiments were expecting a 4-X increase in the number of events to tape. This run had a long period of very successful operations, and, in fact, we delivered record luminosities for an extended period, delivering a total of 5 times more integrated luminosity than in the previous run. However, most of the time was not spent "running."

The main feature of the time line, shown in Figure 1, is the extended periods in which we were had severe problems. The most significant of these was "interval 3" in the Figure. During this time, the Tevatron was incapable of delivering reasonable luminosities because the transverse coupling in the accelerator was quite high. This led to enormous emittance growth and severe current limitations. After several months of intense detective work, including new diagnostic software and hardware, it was determined that a low-beta quadrupole was displaced by a large fraction of a millimeter. The magnet was centered, but it was rolled. Correcting this roll fixed all the problems in a dramatic fashion! The other period, marked "interval 6" in the Figure, was caused by a piece of wire in the Main Ring, which limited the intensity and emittance in the transfers through that accelerator.



Fermilab Run 1B Time Line
1993
1994
1995
S O N D J F M A M J J A S O N D J F M A M J
1
2
3
4
5
6
7
8
9

1. Commissioning of the 400 MeV Linac, the refurbished Booster, Main Ring and PBar Source
2. Commissioning the Tevatron
3. Attempt routine running, but encountered severe problems (in Tevatron) which reduced the effectiveness of the period significantly (see text).
4. Located problem in Tevatron, fixed it and had a period of good, uninterrupted running
5. LN2 Supplier decides to send LN2 to McDonalds instead of Fermilab. Had planned a shutdown for Sept/Oct, so did this shutdown anyway, unprepared.
6. Recovery from LN2 loss, severe problems in Main Ring makes this period less than optimum
7. Problems solved, good running period.
8. Planned M&D Shutdown to install new Main Ring Coalescing cavities.
9. Good running period.

Total time Run 1B: 22 Months = 4 + 3 + 6 + 9.

Figure 1., Fermilab Run 1B time line.

In summary, the time spent during this run was:

Commissioning & Improvements	4 months (18%)
Shutdown & Recovery	3 months (14%)
Run with Severe Problems	6 months (27%)
Operations	9 months (41%)

The distinction between *Operations* and *Commissioning & Improvements* (which includes experimental studies on the complex) is difficult to determine. Here, we use the observation that during the best weeks (of which there were 15), we were able to obtain more than 3 inverse picobarns integrated luminosity at each of the experiments. This normalizes to 13 pb^{-1} per calendar month. Since we provided an integrated 115 pb^{-1} for the run, that would mean that we had the equivalent of about 8.8 months of "good operations" for the run. The rest of the time was spent either explicitly doing non-operational things (shutdown, recovery, beam experiments) or getting up to the 3 pb^{-1} level. Our best two weeks had an integrated luminosity greater than 4.5 pb^{-1}, so even weeks of 3 pb^{-1} had failures and beam studies.*

In these monetarily uncertain times in the USA, the demands placed on our complex by outside forces (e.g., Congress) are increasing while the amount of money we get is decreasing. For example, Fermilab is trying to have the $130 million Main Injector accelerator ready for "Run 2" in 1998. It is likely that this accelerator will be commissioned by the smallest staff the Accelerator Division has had in twenty years, if present trends continue (509 FTE staff in 1982; 458 FTE now; no hiring). Moreover, the time between then and now will be packed with lots of non-operational activities. Thus, the relative lack of operations will continue, and probably get more pronounced, into the next millennium.

6. Conclusions from this Temporal Analysis

The majority of time at a complex like the Fermilab Collider is *not* spent in routine operations. New components are being commissioned for each run (by definition). Thus, we need to become more proficient at commissioning components at all levels. Machine experts must be able to use the most modern equipment, without excessive concern about how to incorporate their component into the Control System. Accelerator scientists must be able to invent new and innovative software methods easily and quickly, and be able to obtain large, time-correlated structures of data from components so that the new components can be commissioned efficiently. It is time to fundamentally incorporate these requirements into the Control System, possibly at the expense of the "stable and consistent view of the complex" which the operations experts have demanded.

7. Implications

In short, the following specifications are the important ones for today's control systems:

1. Control Systems must be flexible enough to adapt to the quickly-changing complex,

2. There needs to be as small an overhead as possible for adding new hardware and new software to the control system,

3. The data acquisition capabilities of the control system need to be enhanced, including:

    3a. More data acquisition bandwidth

    3b. Fundamentally available time stamp on all data

---

* One can recast this in terms of what we *thought* we were doing at the time, and we would have:

Commissioning, shutdown, recovery & general problems	10 months (45%)
Run with Severe Problems	2 months (9%)
Operations	10 months (45%)

4. Commercial products, both hardware and software, should be easy to incorporate into the control system,

5. Computer hardware should be easy to change in order to take advantage of the latest technology; this would apply to the console workstations, front end crates and networks.

6. Applications software should be capable of running under different operating systems. This would be possible if there existed a well-defined application program interface (API) for a control system. Decoupling this API from network and hardware details is essential.

7. Operations experts must accept that their view of the complex is not the only one–their view must coexist with other views within the control system.

8. Specific Recommendations

We are not in a position to make lasting, hard-and-fast specific recommendations on what changes must be made since today's recommendations are out of date tomorrow. Some structural recommendations can be made. It is critical for the Controls Groups to forcefully introduce new computer technologies into service at the earliest possible time. A mature controls group in the present budgetary context tends to not recruit new blood. This makes it even more important to take positive action to ensure that the personnel remain up-to-date and motivated.

There are many promising new features in Control Systems out there which deserve mention. In software and protocols, cdev may work out–the general idea of making a standard controls API is good. TCL/TK and it's derived packages are a very promising, robust systems for doing the windowing interface for all facilities. In fact, TCL/TK have been ported to PCs, so this becomes a platform-independent way to do the GUI. There is a trend towards POSIX-compliant operating systems, which is an excellent basis for guiding future software portability. Our documentation problems may have been solved by the excellent HTML tools ("Web Browsers") available today. And, object-oriented techniques have enormous potential for mitigating controls software complexity at all levels. Recall a nice paper from 91-ICALEPCS [ref: Cork & Nishimure, "Framework for Control System Development"], although many of its ideas are now dated.

In the realm of equipment-level processing, consolidation on the VxWorks operating system has been a great aid to the effort here. We need to continue to use POSIX-compliant OS's, especially the POSIX.4 real time OS definition. There are two nice, forward-looking front-end ideas at Fermilab which deserve attention: MOOC (Minimal Object-oriented Communication) helps the front-end programmer deal with complexity through OO techniques; and the Open Access Front End allows scientist and engineer to measure, model or simulate any aspect of the complex in a robust and convenient fashion by making virtual "front ends" which can calculate a wide range of scientific and engineering quantities on line.

9. Conclusions

Today, we are faced with a complex which is constantly changing. The rate of this change is increasing, but our human resources are shrinking. Also, the users' idea of "capable control system" is becoming more sophisticated. Moreover, the computer-related industries have orders of magnitude more resources for producing stable products than any of us can claim. Therefore, it is necessary to understand and cope with the following consequences: The primary goal of the design of a control system must be flexibility; Control systems must include good data-acquisition capabilities; Commercial products usually should be chosen over home-brewed applications; The view of the complex by operations is going to be less stable.

Adaptation of a Synchrotron Control System for Heavy Ion Tumor Therapy

U. Krause, R. Steiner
Gesellschaft für Schwerionenforschung mbH, D-64291 Darmstadt, Germany

Abstract

At GSI, a tumor treatment facility is being established to demonstrate the therapeutical effectiveness of an improved irradiation method with light ions. This program imposes strict and challenging demands on the operation of the accelerators and hence the control system. These requirements and the means to satisfy them are described.

I. Tumor Therapy

A. Status of Tumor Therapy in Germany

Every year, about 350,000 people out of the 80 million population of Germany develop tumors. 200,000 of them undergo some form of radiation therapy in the course of their disease, often in combination with surgery, or with chemotherapy.

Irradiation with electromagnetic beams is the most common method. In the statistics of successful treatment, about 45% of all cases, radiation therapy is second with 12%, behind surgery (22%). Another 6% of successful cases result from a combination of radiation with surgery. Some tumors, on the other hand, cannot be attacked by surgery, because they are inoperable in general or have developed in or close to risky tissue, e. g. eyes, spinal cord or parts of the brain. Some types of tumor are resistent to electromagnetic radiation [1].

In Germany, about 70,000 patients per year could profit from improved radiation therapy.

B. Irradiation with Heavy Particles

For tumor treatment, light ions (e. g. carbon) have some advantages over electromagnetic radiation, leptons, and mesons. Gamma rays exchange energy with tissue in an exponential function of the penetration depth. This is a problem for tumors deep inside the body. Healthy tissue in front of and behind the tumor is heavily irradiated at the same time.

Protons and heavier ions lose the greater part of their energy at the end of their trajectories, in the Bragg-peak. The penetration depth is a well-known function of energy and tissue density. The tissue behind the end of the trajectory remains largely unaffected and the tissue in the entrance channel in front of the tumor experiences relatively low damage compared to treatment with electromagnetic beams. The damaging effects, double breaks of the DNA-strands (single breaks repair themselves), scale more than linearly with the energy loss per volume, again in favor of low damage in the entrance channel. A treatment split up into about 20 fractions on consecutive days gives the tissue with low damage time to recover while the irreversible damage level in the tumor volume accumulates.

Charged particles can be guided very precisely, and heavy ones do not straggle much. With significant improvements in tumor localization techniques, ion therapy is promising for high-precision treatment.

C. Intensity-Controlled Raster Scan

Particle irradiation is presently used for tumor therapy. The operating proton- (e. g. Loma Linda) and ion- (e. g. HIMAC, Chiba) treatment facilities work with fixed beam energies and without lateral beam deflection. The matching of dose deposition to the individual tumor geometry is achieved mainly by passive elements: sophisticated mechanical masks in front of the patient. The penetration depth of the projectiles is adjusted by the amount of solid material passed by the beam before it reaches the patient. Energy spread and emittance growth inherent with this technique are limiting factors for irradiation precision.

These limitations can be avoided by using an 'active' irradiation technique [2]: A thin "pencil beam" of well-defined energy is scanned over the irradiation volume. The penetration depth of the beam is adjusted by beam energy variation. Every energy step corresponds to a slice of tumor tissue in which the energy loss peaks. The lateral control of the beam is achieved by two scanner magnets in front of the patient. The 'hot' beam spot, some cubic millimeters, is precisely controlled in three dimensions (fig. 1).

A constant scanning velocity would require beams of constant intensity to generate a homogeneous dose distribution. To cope with intensity fluctuations in real beams, the intensity is measured on-line. This intensity measurement is used to control the scanner: the beam spot is shifted to the next position when a predetermined number of ions has been counted. Drop-outs in the beam will simply prolong the irradiation time for the actual volume element.

When the irradiation of one layer is completed, the beam is aborted, and the irradiation is continued with the next beam energy. A treatment session starts with the furthermost tumor layer. All layers — except the most distal one — have to be irradiated inhomogeneously in order to compensate for the effects of beams which have penetrated to irradiate the more distal layers.

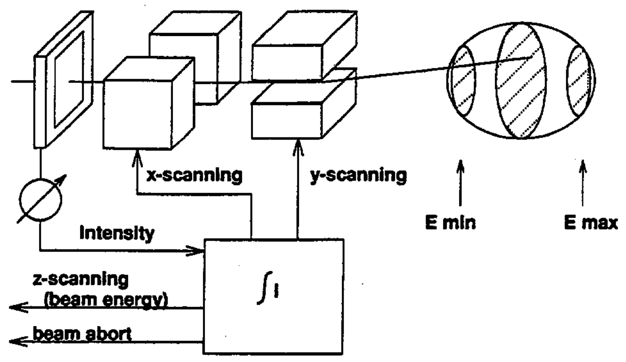


Figure. 1. Intensity Controlled Raster Scan

D. Tumor Therapy at GSI

GSI is a facility for physics research rather than for applications. The linear accelerator Unilac, the synchrotron SIS and the experimental storage ring ESR can handle all ions from Helium to Uranium. For therapy, the synchrotron is oversized in energy by a factor of 4. But presently GSI is the only facility in Europe which can provide all the tools needed for the intensity-controlled raster scan in therapy.

A project has been established to develop a raster scan with novel intensity control for practical use. To verify the usability of the raster scan, and to prove the effectiveness of the therapy, a five year program will start in fall '96, during which about 70 patients will be treated at GSI every year. A long tradition in biophysics research supports the therapy program.

II. Requirements

A. Responsibility of Accelerator and Therapy Staff

The accelerator and therapy facility are under control of different departments:

Operation of the therapy facility is done locally by medical staff. Their responsibility is to handle the treatment plans, to operate the raster scan, to guarantee patient safety and to request the appropriate beams. Sophisticated on-line beam diagnostics are provided to verify the correctness of the beam parameters. In case of intolerable deviations to the treatment plan, the beam is aborted and the irradiation is stopped.

The accelerator staff is responsible for delivering the beams for therapy with an operational reliability as high as possible. The accelerator provides two different means to abort the beam in less than 1 msec, both activated by the scanner control equipment.

B. Accelerator Operation for Therapy

The accelerator has to produce beams of different energies to scan the tumor depth.

The scanner should always operate close to its maximum speed to minimize the irradiation time. Corresponding to the need for inhomogeneous irradiation, beams of different intensities have to be provided. To adapt to different tumor sizes, the beam spot has to be adjustable. All other beam parameters will be fixed. The requirements for accelerator operation are:

- Ion species: $^{12}C^{6+}$ only
- Energy **E**: 80 – 430 MeV/u (20 – 300 mm penetration depth), 255 fixed levels ($\Delta z < 1$mm),
one patient: up to 64 energies
- Intensity **I**: $1{\cdot}10^6 - 1{\cdot}10^8$ ions / spill, 15 fixed levels
- Spot size (focus) **F**: 4 – 10 mm, 7 fixed levels
- Irradiation time for one patient: 5 min maximum
- Preparation time for a patient: 30 min
- Fast beam abort (less than 1 msec)

In order to achieve high operational reliability of the accelerator, the following requirements are laid down for therapy:

- Use only proven and validated settings.
- Protect settings from loss and accidental modification.
- Automatic operation, controlled by scanning equipment.
- Inhibit operator's access during irradiation.
- Avoid networking for settings; provide all therapy beams in stand-by on a pulse-to-pulse request basis.

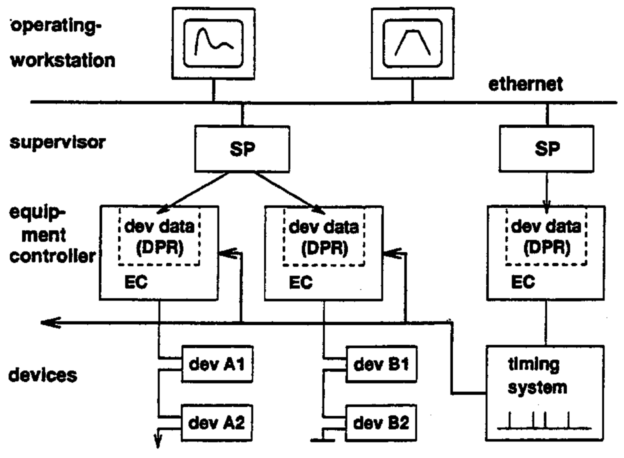


Figure. 2. Control System Architecture

The accelerator can easily provide the appropriate beam properties. But the modes of operation have to be upgraded to meet the requirements for therapy. The main modifications to upgrade the accelerator for therapy are to be made in the control system.

III. The Control System Architecture to Start From

The control system architecture (fig. 2, [3]) corresponds to the so-called 'standard model'. All real-time capability results from actions of the real-time software modules on the Equipment Controlers (ECs), started by the timing system. Communication with the operation level is done by **D**ual **P**orted **RAM** (DPR) on the ECs, the supervisor processors directly read and write the EC data. The real-time software modules run exclusively with data stored in DPRs. No component, hardware or software, from the EC upward is involved in a real-time process.

The DPRs are divided into 16 segments, representing the presently supported 'Virtual Accelerators' (VA). A VA is defined as a complete dataset stored in the DPRs necessary to execute an accelerator cycle. The operations software provides data for the DPRs to create a VA. The timing system makes the VA become 'real', i. e. to execute the datasets, allowing real-time execution of any sequence of VAs.

IV. Upgrade for Therapy

The present controls architecture and its real-time capabilities have met all demands to run the accelerators for the past five years. It is neither desirable nor possible within the given time frame to change it in its basic features. Extensions must be designed as add-ons to the given structure, and are mainly restricted to the equipment control assemblies. The operations software has to remain virtually unchanged. Adaptation of the device control software should be as simple as possible.

A. *Substructuring of Virtual Accelerators*

Pulse-to-pulse switching of the accelerator settings by sequencing different VAs is used routinely during operation of the accelerators, showing that the hardware is reproducible and allows the required therapy beam parameter variation. But the number of VAs is presetly limited to 16, which is insufficient for generation of the therapy beams. The concept of VAs has to be extended, allowing the implementation of enhanced flexibility.

One of the VAs (#15) will be used exclusively for therapy. It will be executed differently from pulse to pulse, representing the settings for therapy, labeled by Energy, Intensity, Focus (E, I, F). Therapy data are stored not in the DPR but on the ECs to allow real-time access. The central timing system announces which of the therapy cycles is to be executed next. Then the data for this cycle are copied to the data area of #15 VA of the DPR (fig. 3). After copying, all device-specific software on the ECs can operate as at present, without any modification.

To store therapy data, we provide flash-EPROM for the ECs to protect against data loss. Special programming sequences, exclusively executed by the ECs, are needed to store data. This sets a high barrier against unwanted data modifications from the console level. New datasets first have to be supplied in the data area of #15 VA and then, after validation, explicitly copied to the therapy memory.

The full range of variable parameters **E, I, F** results in 26,775 different beams. Luckily, any single device depends only on subsets, in worst case on **E**×**F**. According to the device type any dependency of **E, I, F** can be configured, limited only by the available memory.

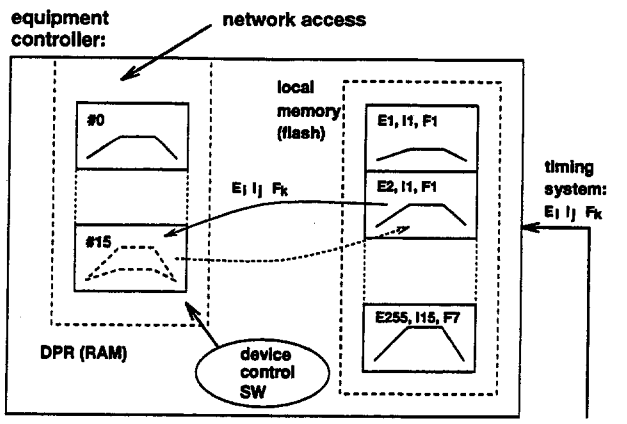


Figure. 3. Substructured virtual accelerators

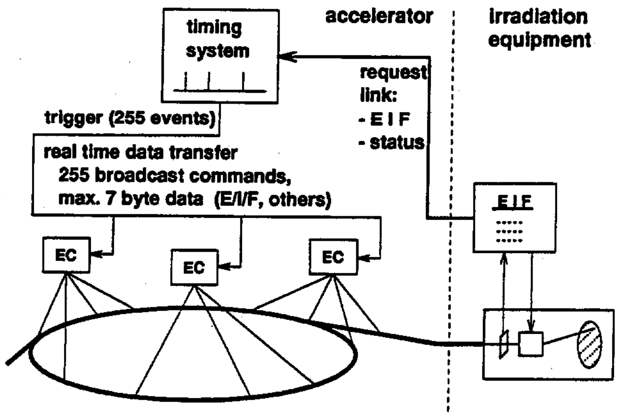


Figure. 4. Request Mechanism

B. EC Memory Upgrade

To store therapy data, about 100 ECs have been equipped with additional flash memory. Previously, EPROM used for the device-control software has been located on a piggy-back. The existing ECs can be re-used by replacing this memory piggy-back by a new one providing 2 MByte of RAM and 2 MByte of flash EPROM. Besides the storage of the EC's program code, it provides sufficient memory for therapy data for the majority of devices.

For devices with big data sets the memory can be doubled to 4 MByte of RAM and 4 MByte of flash EPROM, again allowing re-use of the costly processor part.

C. Timing System

To make the above-described procedures possible, the functionality of the timing system must be extended.

The timing system is the only system-wide interconnection with real-time performance. It can distribute 255 different time marks to all ECs to start real-time software modules at precisely set instants ($\Delta t < \pm 2\mu$sec).

Which beam is to be produced next is determined by the scanning control. Only at the end of each accelerator cycle can the decision be made whether the same beam is requested once more, or the next parameter set is to be executed. To provide the beam parameter information for the ECs in real-time, there is no other means in the GSI control system but the timing system. It is connected to the scanning control by a dedicated hardware link to receive, in addition to some status information, the request for the next therapy beam (fig. 4).

To distribute the beam parameter information (**E**, **I**, **F**) to all ECs, the timing system has been extended with data transport capability. In addition to the time marks, it allows broadcasting of 255 different commands, each with up to 7 Bytes of data. This can be used for real-time command execution on all devices, of which the most important is the delivery of **E**, **I**, **F**.

D. Device Control Software

At present, 45 different types of devices are supported, each of them requiring individual control software. About 20 have to be upgraded for therapy operation.

As far as possible, modifications are done in the system software, common for all devices. This includes data transfer on the timing bus and handling of flash EPROMs. A set of general copy routines is provided, allowing the copying of data between any kind of memory, either RAM or flash EPROM.

Considering device characteristics, device specific software has to be upgraded too. Based on widely-used programming standards, extensions could be written in a general way. Using include files, identical source code is used for all device-specific software. Only a short declaration section is needed, defining basic constants and types to adapt to the peculiarities of device types.

E. Data Supply

Because of the high number of datasets needed for therapy, manual tuning of the machines would be prohibitive. To derive data automatically, a model of the accelerator will be used, allowing calculation of all device data from high-level machine parameters such as ion charge and mass, energy, tune and focusing type.

Data for all therapy beams and all devices are calculated and sent to the devices in dedicated setup sessions. The data are validated with beam to assure correctness and then sealed in the EC's flash memory. Correct operation of the machine will be checked prior to the patient's irradiation. New data supply is expected to be necessary only after modifications to the accelerators or when the modeling of the machine is improved.

F. Accelerator Operation

The preparation time for one patient will be about 30 min, while the actual irradiation time will be less than 5 min. Reserving the accelerators for therapy during the patient's preparation time would result in wasting valuable beam time. The preparation time should be usable for physics experiments. Therapy and physics experiments require different ways of using the accelerators.

For therapy, only tested and approved data and courses can be used. Operation is fully automated; all operator interaction will be disabled during patient irradiation. The only possible human action is abortion of the irradiation by the medical staff.

Physics operation, on the other hand, requires fast and flexible reaction to the various experimental needs. The operations staff has to have full access to all devices to adjust the machines to changed conditions.

To fulfill the conflicting requirements for physics and therapy, the operational mode has to be switched. Switching is initiated by the therapy staff and can interrupt any physics program at the end of a machine cycle.

The command for switching is transferred to the central timing system by the request link. The central timing system then

- disables the execution of all physics cycles (VA #0 – #14),
- sends a lock command via the timing bus to all devices which from then on will reject all operator access,
- sends a command via the timing bus to all devices to verify therapy settings for non-pulsed properties, i. e. cups have to be out of the beam and slits have to be set to predefined values,
- enables the execution of the therapy cycle (VA #15).

Only then can the patient irradiation start: the scanner control requests beams and triggers the beam abort after complete irradiation of one tumor slice.

After irradiation of one patient is finished, the accelerator is switched back to physics operation. The central timing system disables execution of the therapy cycle, enables the execution of the physics cycles and revokes the locking of the devices to allow full access to the operators again. The interrupted physics program will continue automatically.

V. Conclusion and Outlook

With the measures described above, we are sure we can cope with the strict demands of the therapy program in terms of patient security, treatment speed, and treatment precision. Special emphasis is placed on minimizing the patient's stress level. This is why we did not compromise in improving operational reliability and treatment speed.

We hope that the program will be a success in terms of patient healing. If successful, the building of dedicated hospital-based irradiation facilities will be considered. Only with dedicated irradiation facilities can the high demand for treatment be met. For control system designers, it would be a real challenge to design from scratch a turn-key, fail-safe, and fool-proof control system for medical applications.

VI. Acknowledgements

We would like to thank the many people who have contributed to this work, especially G. Englert, W. Panschow and P. Hahne for the equipment controller; W. Rösch and R. Hartmann for the timing system and the request link to the irradiation equipment; L. Hechler (all GSI) and P. Kainberger (EDV-Beratungsbüro, D-64625 Bensheim), for the device control software and many valuable contributions to the architecture upgrade.

References

[1] K. Drumm, *Sozioökonomische Studie zur Strahlentherapie mit geladenen Partikeln*, Ph.D. thesis Heidelberg, 1993

[2] Th. Haberer, W. Becher, D. Schardt and G. Kraft, *Magnetic scanning system for heavy ion therapy*, Nuclear Instruments and Methods in Physics Research A330 (1993), 296-305

[3] U. Krause, V. Schaa, R. Steiner, *The GSI Control System*, ICALEPCS '91, Japan, 1991, KEK Proc. 92-15

CICERO: Control Information systems Concepts based on Encapsulated Real-time Objects

R Barillere[1], E. Bernard[2], C Escrihuela[1], W Harris[3], J-M Le Goff[1], R McClatchey[3]

[1]ECP Division, CERN
[2]SPACEBEL Informatique S.A
[3]Dept Of Computing, Univ West of England

Abstract: Modern High Energy Physics experiments and accelerators require increasingly sophisticated control systems which turn out to be difficult and costly to maintain with the present technology. The situation will seriously worsen in the LHC era. R&D departments of industrial companies are directly concerned with similar difficulties in power plants and complex automated systems. CICERO combines efforts from institutions and industry to address this problem. The main building blocks of a generic control information system have been outlined and constructed. A middleware called Cortex [2] providing the integrating capabilities and the extended communication support for the integration of heterogeneous control products has been developed. To support the long life-cycle of these large projects, the middleware relies on CORBA (Common Object Request Broker Architecture) [1], the powerful emerging standard proposed by all the major computer manufacturers for transmission of objects over networks. In addition, CICERO provides a series of common components, to be used as templates for the construction of specific CORBA-compliant control services such as Archiver/Retriever, Finite State Machines and GUI. A Prototype has been built as a first step towards our main goal of providing technical solutions which could later be the major components of a basic turnkey system for medium-to-large scale HEP experiments and accelerators. The key design constraints and concepts of CICERO are identified in this paper.

1 Introduction

Modern experiments and accelerators at CERN utilise complex equipment for the control of their hardware. The operation of these equipment is managed by increasingly sophisticated control systems software. With the approval of the next generation of accelerators and experiments, the Large Hadron Collider (LHC), this increase in control system complexity continues apace. The technology employed to generate and maintain the LHC beams will require considerable advances in both hardware and software design. The LHC experiments themselves will involve collaborations of many tens of institutes and over 1,000 physicists, engineers and computer scientists from around the world. The knowledge required to construct and monitor parts of the LHC experimental (sub-) detectors will be distributed between these institutes making it difficult to impose standards. There will consequently be significant problems of information transfer to ensure that each (sub-) detector retains autonomy of control but can work with other (sub-) detectors for data-acquisition. In addition, the LHC detectors (both in the accelerator and in the experiments) will be required to have a long lifecycle since the experiments will take data for several years. As a consequence, maintainability will be an important consideration. Furthermore, the accelerator and experimental groups will also be working to very tight time and cost schedules. As the available hardware grows, so the control systems are required to grow from an initial lab-based test system to test-beam operation and ultimately to the fully-fledged accelerator and experimental systems. Control systems designers are becoming increasingly aware that what is needed to facilitate the development of LHC control systems is an integrating framework (or 'software bus') which enables the building of distributed control systems where responsibilities are distributed amongst control elements that have to collaborate together.

Experience of the development of control systems for the LEP experiments [3] and anticipation of the increased demands which will be generated by the new and larger experiments for LHC, has illuminated the main constraints for the design of such an integrating scheme. Firstly, since the LHC control systems will be equipped with typically more than 100,000 sensors and activators, the **management of control system complexity** is a major consideration. Object-oriented design techniques embodying abstraction, inheritance and the use of classes and objects will aid the design here. Secondly the problem of **concurrent and collaborative software engineering** requires addressing. This is particularly true when the development of the control software is carried out by engineers who are separated geographically. Thirdly, the software developed should provide **stability, flexibility and availability** of control system elements so that system down-time (such as that for upgrades) is minimised. Finally, the control system software should provide **balanced, distributed processing** in the heterogeneous environment of High Energy Physics (HEP) control systems.

2 The Problem Statement

HEP experiments and accelerators are normally developed by various teams with specific task assignments. Usually, the different parts of a HEP experiment are developed by different institutes in their laboratories and gathered together later after being tested. The experiments are tuned up at CERN before new tests, or calibration and operation. For instance, the gas system and the high voltage system of a muon detector may be produced and tested by two different teams for ultimate integration in an experiment in the accelerator tunnel.

In HEP, control systems are, by definition, required to maintain equipment in a constant state over the lifetime of the accelerator or experimental data-taking period. Invariably this has led to problems when existing systems need to be upgraded or when new control systems are being specified. Consider these two cases: in the former case, when an existing system is being amended, constraints of time, manpower and cost usually leads to a partial re-engineering of the existing system rather than a complete rewrite of the control system. During the partial re-engineering of the heterogeneous control system, two options are normally considered: either the incorporation of industrial solutions (which can be difficult to adapt) or the investment of considerable in-house manpower to provide equivalent functionality. Other issues that arise with partial re-engineering include hardware and software incompatibilities, lack of commitment to reuse software between control systems and the fact that re-engineering often introduces short- and long-term instabilities with the consequent need for increased human interventions. In the second case when a new control system is being specified 'from scratch', the costs of implementation can be high and with new technology maintenance can be an issue. In this case control systems designers often re-use parts of existing control systems either developed at the laboratory or from industry. To re-use subsets of existing control systems can be a risky strategy since the hardware available for control systems continues to evolve rapidly while integrating industrial control solutions with 'home grown' solutions is often technically infeasible or difficult.

Most of the difficulties identified in amending existing control systems or redeveloping complete control systems from scratch are problems of software engineering. In both cases the software engineering problems are those of taking a bottom-up approach to solving distributed computing issues in heterogeneous environments. Since HEP laboratories are not leaders in computing, designers must make use of industrial solutions and deal with the attendant problems of their integration. Essentially what is required is reusable, distributed and collaborative control systems solutions which enable both partial re-engineering of existing systems and facilitate the construction of new control systems from scratch. These solutions must also minimise maintenance costs and provide highly reliable and deterministic control systems.

Existing control systems software technology has been based on the client-server model of computing using proprietary protocols such as TCP/IP and/or Remote Procedure Calls (RPCs) for communication across the distributed network of control devices. This infrastructure requires peer-to-peer agreement between the collaborating controls partners and cannot provide for dynamic invocation of services. Such systems also suffer from issues of scalability: when the number of clients rises, the load on the server also rises so that system throughout of messages and service calls falls. In addition, using TCP/IP and/or RPCs it is not possible to easily move the server process from one platform to another since the clients must be reconfigured to be made aware of the servers location. Consequently, the user-supplied controls software must be technology independent and free from the confines of the client-server model of distributed computing. The user software must be data-driven and designed to be reused in a control system infrastructure which has been put together using a top-down approach. The user must be able to plug a new controls service into a controls infrastructure with no disruption to the operation of the existing controls systems and with no or minimal recoding.

The challenge is how to build such heterogeneous, distributed and collaborative control systems re-using as much as possible of the existing controls software. One solution which has been proposed in the physics controls community [4] is that of a so-called 'software bus'. In an analogy with hardware, a software bus structure is where different modules plugged into the same bus may cooperate if their bus interfaces adhere to a given standard. A software bus must hide the underlying technology from control systems designers - they have no interest in distributed systems technology nor should they have. It must also scale with the technology used so as to preserve designers' investment and must provide more facilities than simple address and data exchange. In software terms this would translate into agreeing on a common software layer with an unique Application Programming Interface which permits the implementation of shareable software as separate and independent modules. One basis on which to build a software bus is the recent Common Object Request Broker Architecture (CORBA) specification [5] for the implementation of object communication in distributed systems. Such a standard facility supplemented by standard control software, if jointly adopted by laboratories using control systems for physics, could make future controls software more 'shareable' and reusable. The joint adoption of a software bus would be complementary to, not competing with, collaborations like EPICS [6] and products like Vsystem. Examples of applications which could benefit from the use of a software bus in the controls environment include: User Interfaces (development & management system), console manager, on-line help facilities, database access control, realtime data management, data logging, archiver/retriever, alarm handling system, equipment access, system configuration mechanisms.

The following section describes the CORBA specification and illustrates how it can be used as the basis of a software bus. In addition, this section notes where further functionality is required on top of CORBA to satisfy the needs of control systems designers.

3 The OMG Standards

The Object Management Group (OMG) is an industry consortium dedicated to the goal of developing an object-oriented architecture for the integration of distributed applications. The OMG emphasises as its objectives the interoperability, reusability and portability of components and its operating procedures attempt to insure that any specifications adopted as standards have their basis in commercially available software. (OMG, however, only deals with interface standards, not the software itself). The OMG has published several documents including:

- Object Management Architecture (OMA) Guide [7]
- Common Object Request Broker Architecture (CORBA) [5]
- Common Object Services Specification (COSS), Volume 1 [8].

3.1 OMG Object Management Architecture

The OMG's Object Management Architecture (OMA) defines a Reference Model which identifies and characterises the components, interfaces and protocols that compose a distributed object architecture. The four main elements of the OMA are:

- The Object Request Broker (ORB) which enables objects to make and receive requests and responses in a distributed environment. The ORB and related facilities are defined by the CORBA specification [5].
- Object Services - a collection of services (interfaces and objects) that provide basic functions for using and implementing objects.
- Common Facilities [9] - a collection of services that provide general purpose capabilities useful in many applications. Control examples of Common Facilities include Arcghiver/Retrievers and Loggers (see figure 1).
- Application Objects - objects specific to particular end-user applications.

The Object Request Broker in the OMA is viewed as a 'messaging backplane' spanning multiple systems. From the software point of view, the ORB may consist of a single software component, or multiple cooperating (and possibly heterogeneous) software components. The Object Services, Common Facilities and Application Objects in the OMA correspond to different categories of object implementations. In the OMA Reference Model, the purpose of categorising objects into Object Services, Common Facilities and Application Objects is to guide OMG's strategy for developing interface specifications. Object Services include lower-level (and thus the most crucial) object interfaces; and Application Objects reflect the need for independently-developed application interfaces for which the OMG will never develop specifications.

In the OMA, an object is an identifiable, encapsulated entity that provides one or more services that can be requested by a client (which can be another object, or a non object-oriented application). An operation is an identifiable entity that denotes a service that can be requested. The Object Request Broker (ORB) component of the OMA supports cli-

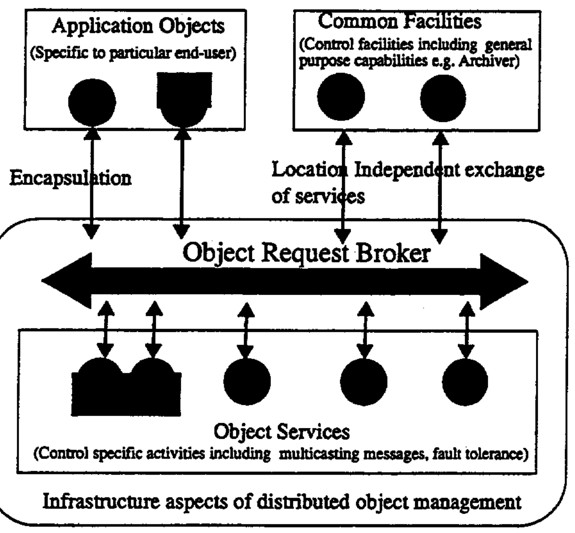

Application Objects
(Specific to particular end-user)
Common Facilities
(Control facilities including general purpose capabilities e.g. Archiver)
Encapsulation
Location Independent exchange of services
Object Request Broker
Object Services
(Control specific activities including multicasting messages, fault tolerance)
Infrastructure aspects of distributed object management

Figure 1: The OMG/CORBA Model for CICERO.
Application objects: Cortex Infrastructure objects.
Common facilities: OODBMS, Real-Time controls, Logger

ents making requests to objects. A request causes a service to be performed on behalf of a client and any results of executing the request to be returned to the client. The ORB is responsible for finding the object implementation corresponding to the target object specified in the request, invoking that implementation, passing it the request for handling and returning the results.

3.2 OMG CORBA

The CORBA specification defines a number of different interfaces (see figure 1). The CORBA specification defines an Interface Definition Language (IDL) for defining the interfaces of objects in [5].

The CORBA IDL plays a role similar to that of the IDL in a Remote Procedure Call (RPC) facility. IDL specifications can be compiled to produce client stubs and implementation skeletons (essentially server stubs). The use of a compiled client stub to access an object interface is referred to as the static interface. A Dynamic Invocation Interface (DII) is also provided that allows clients to construct requests at run-time to objects whose types might not have been known at compile-time.

Objects are made known to the ORB by being registered with an object adaptor. The definition of a Basic Object Adaptor (BOA) is contained in CORBA. The BOA is generally designed to handle objects that are independently constructed, and must be handled individually by the adaptor. The CORBA specification also identifies other types of adaptors that might be more appropriate for objects implemented in different ways. For example, CORBA identifies a DBMS object adaptor that would be more appropriate for objects defined within a database environment.

3.3 OMG Object Services

An Object Service defines interfaces and sequencing semantics that are commonly used to build well-formed applications in a distributed object environment. Each Object Service provides its service to a set of users. These users are typically Application Objects or Common Facility objects that, in turn, provide support for specific application domains like network management or complex systems controls. In non-object software systems, a system's Application Programming Interface (API) often is defined by a single interface. The OMG Object Services API is modular; particular objects may use a few or many Object Services. By being object-oriented, the OMG Object Services API is extensible, customisable and subsettable; applications only need to use services they require.

The operations provided by Object Services are made through the CORBA IDL or through proposed extensions to IDL compatible with the OMG Object Model. However, while OMG requires an IDL interface for each object service, implementations of object services need not themselves be object-oriented. Such implementations may continue to support non object-oriented interfaces for compatibility with an existing product's API or with non-OMG standards, in addition to an IDL interface. Also, objects need not use the implementation of basic operations provided by Object Services, nor must objects provide all basic operations.

Various Object Services have been produced at various times by OMG. Examples include Event Notification, Lifecycle and Naming Services which are specified in [7]. A Persistence Service specification has been approved and will appear in a forthcoming update of [7].

3.4 Implications of Using CORBA in Controls Environments

Any software bus to be used in the controls environment that is based on CORBA can thus provide standard information exchange between controls devices as well as hardware independent services. Controls systems users can use this integrating infrastructure to provide services across the control system and potentially between control systems. Since the infrastructure is based on CORBA, it will be both scalable and reconfigurable and therefore a control system could be optimised to handle on-line loads.

CORBA provides 'plug-and-play' at the object level and the possibility of moving controls objects around the control system without stopping the client applications. Further it facilitates network independence and can run TCP/IP or RPCs on networks such as Ethernet, ATM or SCI. Adherence to OMG standards such as the Common Object Services, (COSS), provides facilities such as naming and authentication as well as object adaptors such as that available for OODBMS access.

CORBA then provides much of the functionality required as a basis for a software bus in physics controls systems. What it fails to make provision for is message multicasting, service invocation/handling support, fault tolerance and object management aspects specific to the domain of control systems. Additional software is therefore required alongside CORBA to complete the software bus and thereby to provide the functionality demanded by control systems users. The CICERO project has created Cortex [2] to provide this controls-specific software and its design concepts are explored in the following sections.

4 Cortex Design Concepts

The Cortex element of the CICERO project provides the software bus [2] which runs on top of CORBA. It enables the building of distributed control systems where responsibilities are distributed amongst control elements that have to collaborate together. Cortex supplies extra features on top of CORBA: muticasting of messages and service handling in addition to a series of control services (archiver/retriever, logger, state machine, distributed user interface) which could be standardised in the framework of OMG.

In addition to the functionality already identified, Cortex promotes the modularity of distributed control systems in order to allow some integrated control systems to be operational while some others are not. Cortex will also support users during the reengineering phases of the integrated control systems by allowing them to reuse and integrate existing software.

Cortex supports the entire collaborative distributed control system life cycle by:

• providing a multi user development environment,

• supporting the testing and simulation of users' control elements with respect to the integration within the Cortex control system,

• offering tools to operate and protect the control system from faulty processes implementing some control elements,

• allowing the integration of already existing control systems,

• allowing collaboration with other autonomous control systems.

Cortex is intended to provide a control system designer with the ability to integrate his particular control system efficiently and with minimal cost and effort.

The architecture of any distributed control system will change during the lifetime of the accelerator or the experiment it is operating. The control system integrating platform, Cortex, must therefore support a mechanism to allow a new version of the distributed control system to be in preparation while an older one is operated on-line. It shall also support the backup and the restore of a given version of the collaborative distributed control system. Furthermore, tests and validation (and possibly simulations) of a new configuration will be needed before it is applied to the operating on-line system.

These constraints have led to a so-called dual-face approach (see figure 2.) being taken in Cortex:

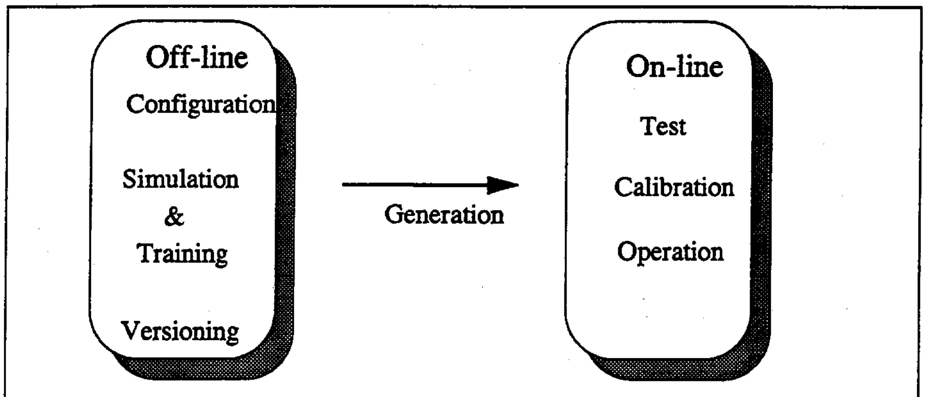


Figure 2: The Dual-Face approach

• an off-line Cortex representation is required to handle the logical descriptions of the architecture of the distributed control system and to describe the various information and commands to be exchanged between the different control elements. This so-called *Repository* also holds the description of the hardware model from which the on-line distributed control system is constructed, and

• an on-line Cortex representation is also required through which the control elements can exchange information and commands in a pseudo- 'plug-and-play' fashion. Control elements operating within the Cortex Infrastructure can access the Cortex Repository through this so-called *Infrastructure.* A generation mechanism is provided to facilitate updates of the on-line Infrastructure according to the Repository contents.

The responsibility of Cortex is twofold: on the one hand it has to support the description of the architecture of the distributed control system and the definition of the information and commands to be exchanged between the control elements. The off-line representation of the control system therefore addresses the issues of the **management of control system complexity** and that of **concurrent and collaborative software engineering** identified earlier. On the other hand, Cortex must transport and distribute these data and commands to the appropriate control elements when part or all of the distributed control system is in operation. This distribution must be independent of the number of hardware elements on which the various control elements are operating. The integrating framework must be flexible enough to support the addition or removal of control elements, without deteriorating the operation of the rest of the distributed control system. The on-line Cortex Infrastructure thereby addresses the demands of **stability, flexibility and availability** of control system elements and that of providing **balanced and distributed processing** for the control system. Cortex is not a product in the sense of FactoryLink but rather a resolution of a set of issues specific to the controls world which may point the way ultimately towards the establishment of a real controls standard.

5 Cortex off-line

Off-line a Cortex control system is visualised as a collection of collaborating components. These components map onto those on-line processes in a control system which produce or consume information and they can be of various kinds. They may or may not be "control components". For example gas systems or high voltage systems are managed by software that are control components. Each has a read-out system and has responsibility for the hardware. Indeed some control components may need real time capabilities, for instance, elements interfacing front-ends which perform interlocks. On the other hand, an on-line documentation element or a user interface element are "non-control elements" - they do not have responsibility for the hardware. They may, however, support high level control functionalities such as alarm filtering, user assistance, preventative maintenance etc. Cortex can therefore be used to integrate software which implements facilities common to any kind of control element such as loggers, archivers, retrievers, GUIs, final state machines, etc.

Compositeness is the mechanism proposed in Cortex to support the logical encapsulation of a distributed control system. Components may be composed of smaller components. Such components are referred to as composite components and are often used to separate functions or to provide the granularity required by the underlying hardware (figure 3). Users must be able to operate the complete control system from a global standpoint or operate each component independently via the composite components. Additionally, users must be able to specify communication requirements at any level of component, regardless of the inherent hierarchical organisation.

Grouping is a complementary mechanism to Compositeness proposed in Cortex to allow specification of information and command exchange at any level of granularity. A collaboration group is composed of a set of components that make available certain information and services to the other components in the group. Two components will be able to exchange information and commands if and only if they belong to the same collaboration group. Components can be part of more than one group. Collaboration groups can be established across encapsulations of sub-systems. The combination of compositeness and collaboration groups allows the user to refine and optimise the communication at an appropriate level of control system component.

Within a collaboration group, a component can provide information to other components by publishing items (data or services). If granted permission by the publisher, any component of a collaboration group (other than the publisher) can access this information by subscribing to the published items. The set of published and subscribed items handled by a component within a collaboration group is called a component interface. A component usually has a different interface for each collaboration group in which it is a member.

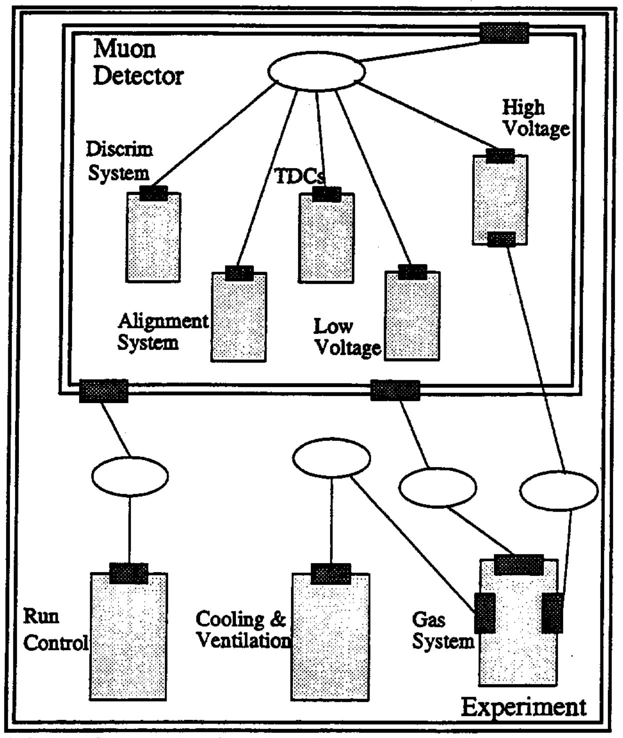


Figure 3: (Composite) Components & Grouping
Key: Groups are represented by ellipses, Components by rectangles and Component interfaces by shaded boxes. Composite Components are shown as double rectangles.

Compositeness and collaboration grouping are the mechanisms for supporting information abstraction in Cortex: some of the collaboration groups can be organised to enforce encapsulation of sub-systems (so-called *strict encapsulation*). This will allow composite components to subscribe within their encapsulation and publish refined information to other composite components in other collaboration groups.

6 Cortex on-line

The Cortex on-line Infrastructure is a set of entities responsible for the distribution of information and for the transmission of service invocations to the appropriate components, according to the model in the Cortex Repository.

The Cortex Infrastructure provides two selectable ways of data exchange between components. The *push mechanism* which allows components to push new information into the Infrastructure or to receive information from the Infrastructure and the *pull consumer mechanism* which allows components to retrieve information from the Infrastructure at their convenience. The push mechanism is recommended for security information such as alarms. The pull mechanism is more suitable for monitoring components offering refreshed information upon user request. In both cases, only information specified in the Cortex Repository will be transported and delivered to the appropriate components. Version inconsistency between the information sender and the information receiver are handled by the Infrastructure at message level. For example, a component may pull from the Infrastructure items which are no longer published (because of modification, replacement or deleteion). The dynamic information contained in these items is no longer refreshed and the corresponding component will be informed. Time stamps will contain the last time these items have ben refreshed.

The on-line architecture supports a separated set of messages to handle service invocation called command messages. Commands are persistent in the Infrastructure from the moment they have been issued until they are completed (successfully or not) or refused by performers. Two basic types of services are available. Firstly, services can be cancellable or non-cancellable: a requesting component can cancel its request while the component offering the service is processing the command. Secondly, services can be multi- or single-requestable: more than one requesting components can issue a command to the same performer component.

More than one requesting component can invoke the same single-requestable command hence addressing the same performing component. The on-line architecture handles possible access conflicts using an internal protocol based on locking. Commands are not direct implementations of operations in the OMG Object Model, which do not support some specific control functionalities such as authentication, availability and progress report features.

7 Mapping OMG Standards onto the Cortex Design Concepts

In Cortex, an ODBMS is being used as the vehicle for the off-line Repository to support a standardised access for the CORBA objects in the on-line Infrastructure. The Repository has been designed to support the notions of Compositeness and Collaboration Groups, Publishing and Subscription and Components as described in section 5. ODBMSs provide persistence of object information, and all the advantages of DBMS systems such as version management, concurrency control, security and recovery. This enables control system designers to save a full description of the experiment or accelerator setup in an object base and to modify that description over time as the equipment grows.

The Cortex on-line Infrastructure is instantiated as a set of CORBA objects responsible for the distribution of information and for the transmission of commands to the appropriate components, according to the description resident in the Cortex Repository. On-line objects in Cortex are written in C++ and use Iona Technologies implementation of CORBA, called Orbix, for object location, access and communication services. The physical location of the components and the Infrastructure will depend on the hardware setup available. This setup may evolve with time for performance reasons or for maintenance purposes. In these cases, the system functionalities must be maintained when part of the hardware is changing. This operation should take place without disturbing the operation of the parts of the distributed control system which are not involved in this upgrade. The location transparency is fully supported by CORBA. CORBA makes no provisions for message sender identification. Any program can potentially send a message to a CORBA object. To avoid unpredictable overloads, Cortex provides an authentication mechanism to ensure that any new starting process is effectively representing a component known to the Cortex Repository.

As an example of the use of Cortex + CORBA, consider the case cited by Clausen [4] at the last ICALEPCS conference where a data logging system is required which allows data to be stored in a circular buffer, later in a database and finally on tape. From the operators console, data needs to be retrieved from all three sources (buffer, database & tape). Individual solutions exist for each form of data storage (EPICS, ORACLE, TapeManagers). The problem is providing a facility that irons out the differences between accessing these three different technologies and allows the operator to access the data regardless of the underlying technology. The software bus of Cortex + CORBA can provide the solution. Adherence to the CORBA standard enables independently defined (client and server) applications to be specified. All interaction between these services is then handled by the Common Object Request Broker. Furthermore the Cortex layer of software then deals with all issues directly related to inter-connecting controls applications and their inter-operation.

As an example of the use of the OMA standard in Cortex the next section shows how Cortex provides reusability both of control system components and complete control sub-systems.

7.1 Reusability and Cortex

The basis for re-use in the CICERO project is through the use of CORBA IDLs and ORBs. Reusability can be exploited in CICERO both at a level internal to components and at a level external to components. Firstly at the internal level, consider the re-use of component code as the hardware of the control system is evolving. By using IDL stubs for client invocation and an IDL skeleton to package the existing (component) code for CORBA compliance, the control system can be allowed to grow whilst reusing existing components. As an example, consider the incorporation into a UNIX-based control system of an existing data logger which runs on VMS/ORACLE. To reuse this component it is necessary only to build an IDL interface to this logger, using an ORB which supports VMS. Then a UNIX component will be able to send messages such as store and retrieve through this interface to log information without having to know the complexities of UNIX-VMS translation. In addition, such a component will be able to log any additional Cortex messages issued by any other existing components according to the description stored in the Cortex Repository.

Using CORBA IDL for interface specification permits the sub-division of a large software module into smaller, easier-to-manage units with simpler functionality. This facilitates reusability in that it supports:

• the evolution and partial upgrade of complex control systems and

• the re-use of existing (legacy) systems through CORBA objects.

In this example of reusability, partial re-engineering of component code is again required since the IDL specification takes place inside the component code.

CICERO is also able to exploit reusability at a level external to components. This can be achieved through the use of so-called Reusable Components. These can be regarded as templates for components with logical input/output which can be instantiated as many times as there are physical devices. At the time of instantiation, there may be no hardware assignment - only at the time of assignment will the desired functionality become apparent and the component code reused. For example, consider a 16-channel Analogue to digital Converter (ADC) read-out component coupled with an ADC Converter component. The first component is hardware independent, except for the ADC gain, and can be reused as many times as ADCs are needed in the system. The second component is context dependent and can evolve with the hardware of the system. If the second component is data driven and obtains its configuration data from the Repository, then the ADC/ADC Converter pair can be reused with no code modification in any subsystem of the experimental setup. Cortex offers the possibility of decomposing a complex control system into data acquisition components, command components and automation loop components offering the possibility of reusing or sharing components in different control systems.

It is also possible in CICERO to sense whole sets of components at the control system level. Consider two Cortex control systems developed independently and merged at a point in time. For example, a development or test setup which has been locally set up and then physically moved to be made part of (integrated with) a larger control environment. In this case since there is compliance at the Cortex level, no code modification whatsoever is required and the integration takes place through the addition of an intermediate component which resolves the match between the two Cortex systems. That is, the intermediate component subscribes to the items of the test control system and republishes the converted items for the larger control system. In addition, it is possible to split the two control systems for independent running at a later point in time without code modification.

8 Conclusions

The main conclusions that can be drawn from the experience of the use of OMA and CORBA are that:

• Cortex has successfully demonstrated the ability to allow control systems designers to wrap up existing legacy software systems, some of them shell scripts, as CORBA objects and integrate them into the Cortex system.

• the CORBA interface inheritance has allowed reuse of object behaviour across control systems.

• considering that the collaboration team is spread across Europe and beyond with each group working on its own part of the prototype, the "plugging" together of the components using Cortex + CORBA to construct the prototype was a notable success.

• none of the OMG fundamental object services (COSS) have been implemented as yet, in fact most of them have yet to be fully specified. This presents a dilemma as the CICERO team has proceeded to the design stage and have commenced specifying and designing its own object services. It will be interesting to see how long it takes for the OMG object services to be bundled into the existing commercial CORBA Orbs.

• similarly the OMG Common Facilities Architecture has only recently been published. However this document is intended more as a management tool intended for controlling development of and positioning Common Facilities and their specifications. The experience of using the OMG OMA within the CICERO Project may well provide the basis for the collaboration team to submit suggestions to the Common Facilities Task Force as to candidate common facilities within the control systems vertical market.

9 Acknowledgements

In a collaborative project such as CICERO RD38, each collaborator deserves the thanks of the authors. Rather than cite each contributor, thanks are extended to members of RD38 from BARC (Bombay, India), CERN (Geneva, Switzerland), CIEMAT (Madrid, Spain), IVO International (Helsinki, Finland), KFKI (Budapest, Hungary), OBLOG (Lisbon, Portugal), SEFT (Helsinki, Finland), SpaceBel (Brussels, Belgium), UID (Linkoping, Sweden), USDATA (Dallas, USA), UWE (Bristol, UK), Valmet Automation (Tampere, Finland) and VTT (Oulu, Finland). Special thanks go to those involved with the development of Cortex.

10 References

[1] J. R. Rymer, 'Common Object Request Broker - OMG's New Standard for Distributed Object Management', *Network Monitor* Vol. 6, No 9, pp 3-27 (1991)

[2] R. Barillere et al., 'The Cortex Project: A Quasi- Real-Time Information System to Build Control Systems for High Energy Physics Experiments', *NIM A* **352**, pp 492-496 (1994)

[3] R. Barillere et al., 'Ideas on a Generic Control System Based on the Experience of the Four LEP Experiments Control Systems', Proceedings of the ICALEPCS '91 Conference, Tsukuba, Japan pp 246-253 (1991).

[4] B. Kuiper et al., 'Panel Session on Software Sharing' at the ICALEPCS conference 1993, *NIM A* **352**, pp 501-515 (1994).

[5] CORBA: 'Common Object Request Broker Architecture and Specification (CORBA)', Revision 1.2, Object Management Group Inc., OMG TC Document 93.12.43 (1993).

[6] L.R. Dalesio et al., 'The Experimental Physics and Industrial Control System Architecture: Past, Present and Future', *NIM A* **352**, pp 179-184 (1994)

[7] OMG: 'The Object Management Architecture (OMA Guide)', Revision 2.0, Object Management Group Inc., OMG TC Document 91.11.1 (1992).

[8] OMG: 'Common Object Services Specification (COSS) Vol. 1', Revision 1.0, Object Management Group Inc., First Edition (1994).

[9] OMG: 'Common Facilities Architecture (CFA)', Revision 4.0, Object Management Group Inc., OMG TC Document 95-1-2 (1995).

The Engines that Drive Collaboration

Rick Stevens
Mathematics and Computer Science Division
Argonne National Laboratory

Abstract

The emergence of virtual reality technology has introduced an exciting new approach to scientific research and development. Virtual reality environments such as the CAVE (Cave Automatic Virtual Environment) are allowing researchers to conduct three-dimensional simulations---to go "inside" experimental data and interact with results computed on high-performance supercomputers. Current applications include drug design, simulation of the casting process for car and aircraft parts, and modeling of combustion within commercial boilers. In the future, collaborative virtual laboratories may enable distributed scientific research groups to share unique large-scale experimental facilities such as accelerators, electron microscopes, and light sources. Using emerging high-speed networks like those being developed as part of the I-WAY (International Wide Area Network) project, researchers in different locations will be able to collaborate naturally and effectively over rich virtual environments that support audio, video, and gesture capabilities. Through telescience technology, researchers also may remotely control operations in deep-sea exploratory submarines, space shuttles, and nuclear reactors. Virtual environments present significant technological challenges, but the rewards are clear: new insight; faster understanding of complex, multidimensional phenomena; and increased scientific productivity. Noteworthy examples are the I-WAY project, which focuses on mechanisms to link supercomputers and advanced visualization environments via asynchronous transfer mode; and the LabSpace project, exploring electronic virtual laboratories to support remote operation and control of scientific experiments.

(The author did not provide a formal written version of his paper but instead made available the transparencies from his talk.)

The Engines that Drive Collaboration

Rick Stevens

Mathematics and Computer Science Division

Argonne National Laboratory

http://www.mcs.anl.gov/people/stevens

stevens@mcs.anl.gov

What is the talk about..

- Long term vision for collaborative environments
- Computing and networking technology
- Research projects and directions
- Connections to the I-WAY project
- Open problems and Invitations for collaboration

Roadmap for Research Programs to Support the Global Information Infrastructure

Collaborative Modeling

today's Grand Challenges

Algorithms

- physical models
- matrix methods
- spectral methods
- reactive flows
- molecular models
- electronic structure
- structural mechanics
- n-body methods
- optimization

Architectures

- workstations
- clusters
- single MPP
- vector computers
- simple distributed sys
- largely single domain
- strong on Flops
- weak on I/O
- small data systems

Tools

- compilers
- debuggers
- auto parallel pre-X
- event tracing
- visualization
- algorithm prototyping
- libraries (MPI, p4)
- "tool kits"
- templates

Networks

- ethernet (LANs)
- FDDI (LANs)
- HIPPI (sub-LANS)
- FCS (not yet)
- ATM (not yet)
- fiber infrastructure
- long haul T1/T3
- experimental OC-48
- workstation peers

Languages

- Fortran 77
- C
- C++
- HPF (not yet)
- Fortran 90 (not yet)
- Fortran M (not yet)
- PCN
- Linda
- extensions vs new

Applications

- extensions of vector codes
- physical models
- few non-science
- few engineering
- mostly modeling not design
- little linkage to exp
- batch still prime mode

Collaborative Analysis and Design

distributed shared resources

Algorithms

- caching, browsing
- MDA, MDO
- statistics
- comparative analysis
- interactions
- CAD linkage to models
- database linkage
- compression
- automatic filters
- neural networks

Architectures

- workstation aggregates
- MPP aggregates
- heterogeneous nodes
- cooperative systems
- mobile nodes
- database linkages
- multimedia support
- archival storage
- data systems
- I/O intensive

Tools

- multi-user AVS
- multi-user MMA
- hetero MATLAB
- linkage CAD/MMA
- linkage CAD/MPP
- linkage CAD/DB
- linkage CAD/MDA
- multisite above
- penbased interfaces

Networks

- gigabits to the desk
- national filesystem
- ATM combo
- nonmetered
- bandwidth on demand
- high peak / low mean
- aggregate for blasting
- standard HW/SW
- blasting protocol

Languages

- integrated analysis and CAD languages
- MDO extensions
- hetero parallel
- fast MATLAB
- AVS like extensions
- OBJECT support
- implicit I/O support
- DB integration
- analysis objects

Applications

- manufacturing of complex systems
- analysis of 3ary science problems
- rigorous analysis of computational exp
- simulation intensive engineering

Rick Stevens, Mathematics and Computer Science Division, Argonne National Laboratory

Virtual Reality and Multimedia

multiviewer interactive distributed deployment

Algorithms

- projective geometry
- rendering
- kinematics
- linkage to physics
- new paradigms for visualization of data
- interactions
- compression
- indexing
- data movement

Architectures

- CAVE^n
- CAVE++
- HD-CAVE
- HD-Goggles
- body suit I/O
- retinal imagers
- 3-D projection
- holographic display
- holographic input
- mobile links

Tools

- 3-D AVS
- 3-D highend MMA
- multisite CAVE env
- 3-D tcl scripting
- CAVE/ MOSAIC
- Goggleland
- connections
- linkages
- pad based kibitzing
- scalability

Networks

- CAVE-to-CAVE
- low latency sync lines
- scaling up/down
- 10's Gb each site
- fanin/fanout problem
- GB to each user
- mobile head to head
- LAN-MAN-WAN

Languages

- 3-D and VR extension
- VR objects
- parallel objects
- parallel interacting agents
- coupling to world DBs
- visual programming
- visual tuning
- pen based sketches
- 3-D story boards/lang

Applications

- medical imaging
- remote exploration
- database mining
- education
- telecollaborations
- 3-D data immersion
- micro manufacturing
- entertainment
- travel reduction

Telepresence

remote manipulation and monitoring
remote "participation"

Algorithms

- autonomous systems
- solids models
- planning, interactions
- navigation
- multimedia fusion
- sensor fusion
- human/computer int
- compression
- geometry reasoning
- vision and mages

Architectures

- gigabits network
- hetero drivers
- MPP
- workstation
- realtime links
- multi-sensor
- robotic support
- manipulators
- nanobots
- mobile support

Tools

- protocols
- sockets for multisense
- SVP simple video pro
- roboKIT interfaces
- stationary video interf
- voice/vid/window
- manipulator KIT
- multiview protocol
- DB interfaces
- agent interfaces

Networks

- gigabits/human-rin
- short range radio
- transparent LB/radio
- digital video
- low latency ATM
- radio ATM interfaces
- multimedia mix
- multisense mix
- network compressio

Languages

- scripting for RINs
- manipulationscripts
- programming by example
- multisense objects
- multisense extensions
- linkage to multisense
- databases
- autologing of events
- autolog I/O library

Applications

- monitoring r activitie
- r manufacturing
- r assembly
- r exploration
- r experiment manip
- r inspections
- semi auto develop
- r piloting
- toys

Rick Stevens, Mathematics and Computer Science Division, Argonne National Laboratory

Workthink Trends

- Virtualization of organizations
- Merging of work/life/play/home/school
- Work as conversation/communication
- Dominance of new/multidisciplinary problems
- Dominance of consumer products/tech drivers
- Self-assembly (business, research, education, recreation)

What is the right infrastructure to support future work?

How would life be different if travel was free and instantaneous ?

Hmmm ?

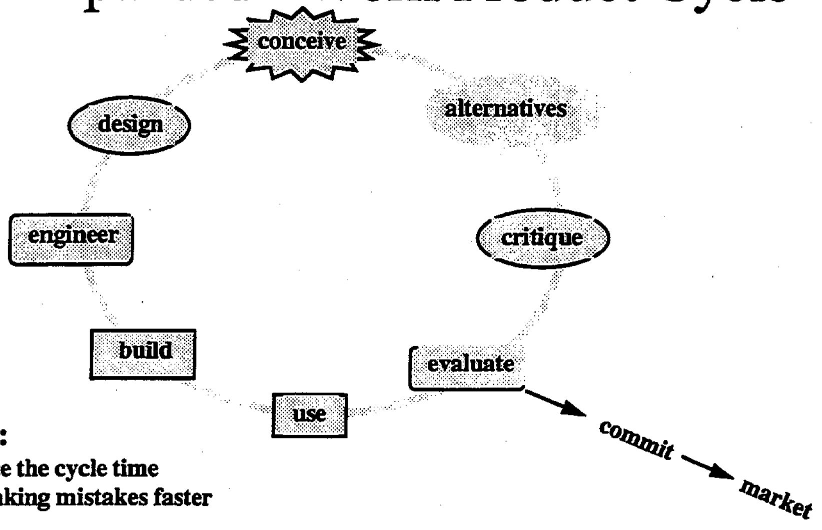
- You would live **where you want to.**
- You and your colleagues would meet in **interesting places.**
- You would **leave your things** where you need or use them.
- You would need to **"arrange"** to meet people in certain places at certain times.
- You would need mechanisms to **"locate"** people and things.
- **Security** would be problematic. (see Tales of Known Space)
- You would develop **habitual hangout places**, most likely to find friends and family.
- You would adjust. (with apologies to **John Gorka**)

The Need for Persistent Electronic Spaces

Adding the concept of **Persistent Electronic Spaces** to the current suite of computer supported collaborative work tools can provide the virtual equivalent of instant and (almost) free travel.

- Persistence is needed to build electronic communities
- Persistence is needed to map the real world to virtual environments
- Persistence is needed to lead us away from the phone call model for collaborations towards the "**Cafe**" model
- Persistence is needed to improve navigation and discovery processes for the GII

Expanded - Work/Product Cycle



goal:
reduce the cycle time
by making mistakes faster

Virtual Organizations:

instant teams
spontaneous workgroups

- Have no particular physical location
- Can have mission oriented lifetimes
- Can have non-hiearchical structure
- Require awareness and loyalty of members
- Require a variety of communications support
- Require sharable resources (data, machines, etc.)
- Require mechanisms for organizational navigation and discovery
- Require resource allocation mechanisms
- Require ability to determine membership status

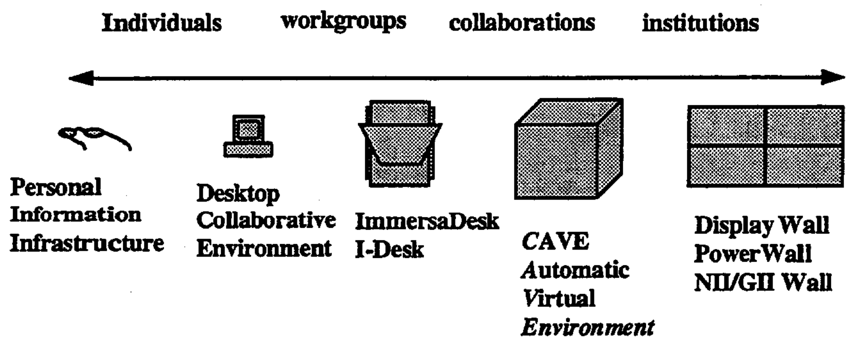
What do we Mean when we say Virtual Environments?

- 3D real time Graphics
- Point of view rendering
- Wide field of view (immersive)
- High resolution
- Audio and other sense modalities (touch, temp, smell, etc.)
- Responsive model interactions (low lag)
- Collaborative (i.e. ability to interact with other people & VR at the same time)

Collaborative Environment Scale Metaphors

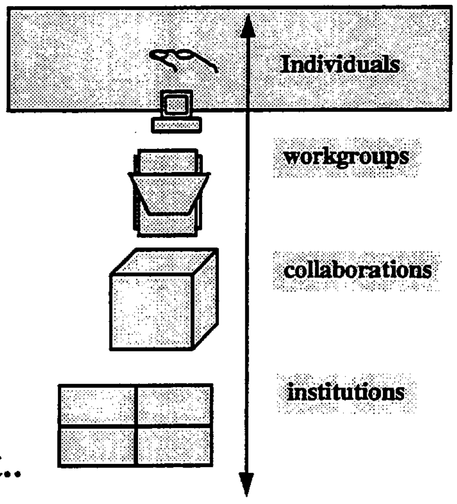
Personal	**Desktop**	**Desk**	**Room**	**Theater**
wrist watch	tv	drafting table	library	movie house
earrings	stereo	sewing mach	darkroom	church
pocket books	cd-rom	piano	kitchen	store
pens	sketch pad	table saw	wood shop	mall
eyeglasses	PC	cooktop	art studio	sports arena
clothing	keyboard	sink	garage	concert hall
cell phone	mouse			
headsets	telephone			
PDA	**MediaStation**	**ImmersaDesk**	**CAVE**	**DisplayWall**

Continuum of Collaborative Environment Display Technologies


Individuals
workgroups
collaborations
institutions
Personal Information Infrastructure
Desktop Collaborative Environment
ImmersaDesk I-Desk
CAVE *Automatic Virtual Environment*
Display Wall PowerWall NII/GII Wall

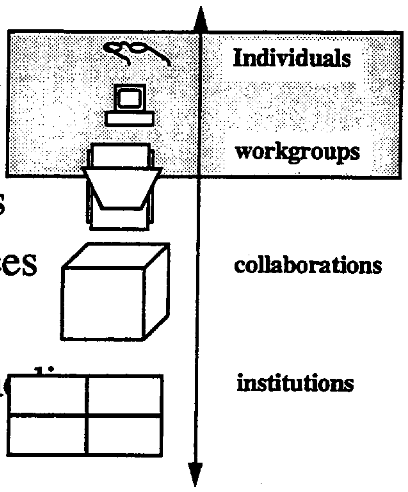
Personal

- Lightweight
- Wireless
- Wearable
- Always available
- Audio dominate
- Keyboardless
- $80-$800 price point..



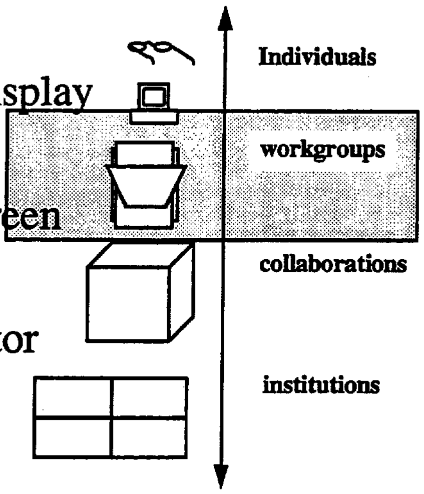
Desktop

- Interactive video
- Multichannel audio
- Text, 2 and 3D graphics
- Flexible drawing surfaces
- Keyboards/mouse
- Ambient and focused m
- Multiscreen
- $8K price point



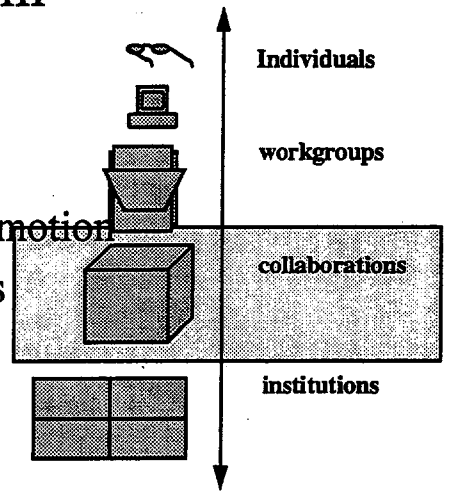
Desk

- 3D immersion/stereo display
- User tracking
- One high-resolution screen
- Seated or standing
- Drafting table form factor
- Stereo Audio
- 1-3 people
- $80K price point



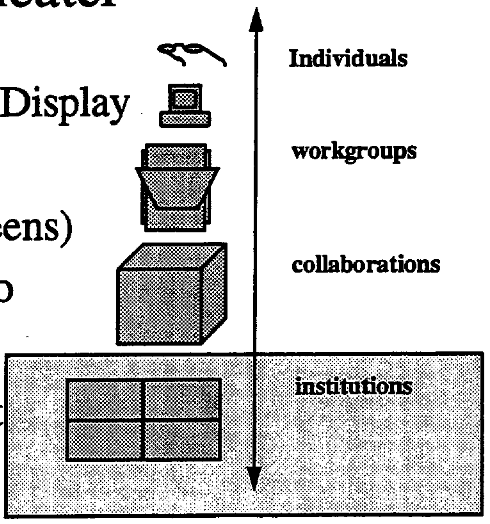
Room

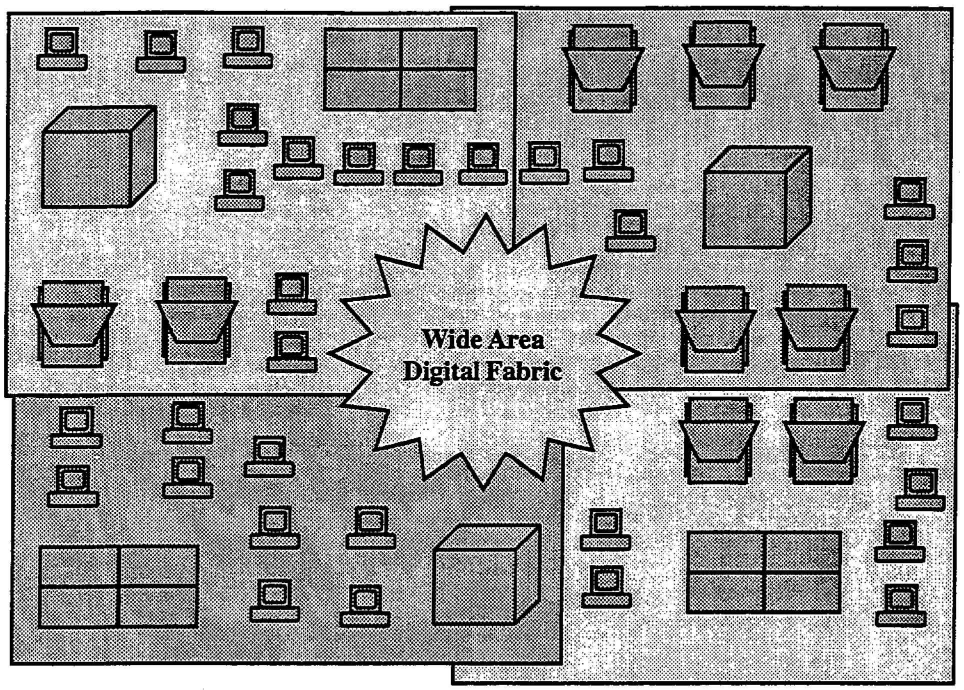
- 3D immersion
- 1 user tracking
- Standing with limited motion
- 4 high-resolution walls
- 1-8 people
- Quadraphonic Audio
- $800K price point



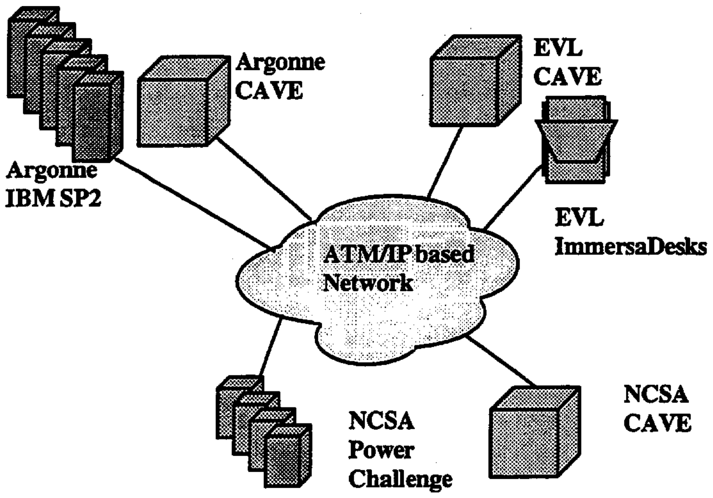
Theater

- Limited 3D Stereo Display
- Large format
- (NxM high-res screens)
- Multichannel Audio
- 10-100's people
- $800K+ price point



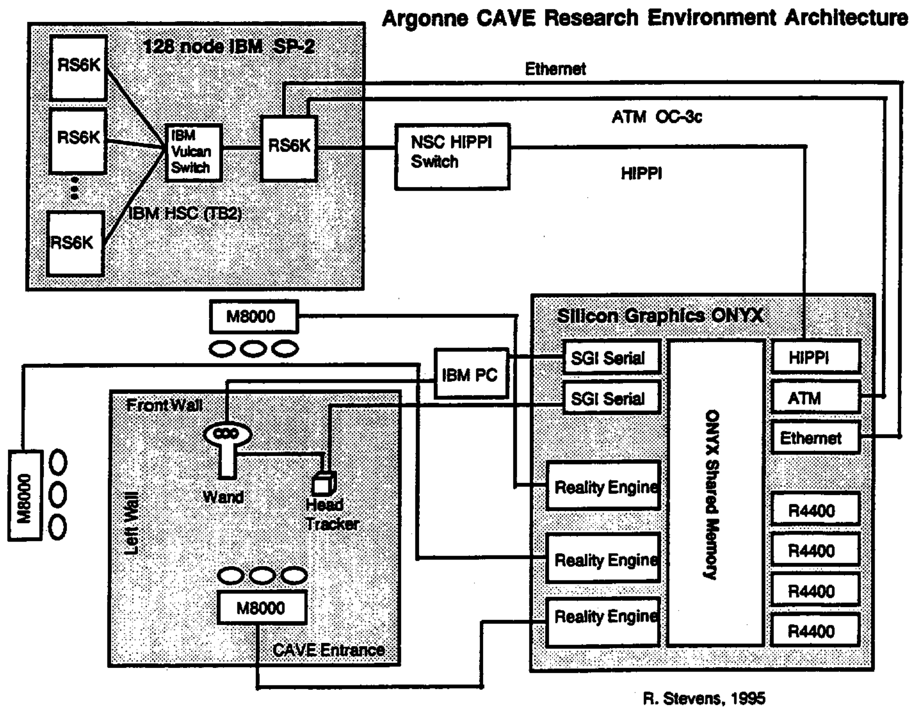


I-WAY Testbed Alpha'



Integrating Virtual Environments and Massively Parallel Computing

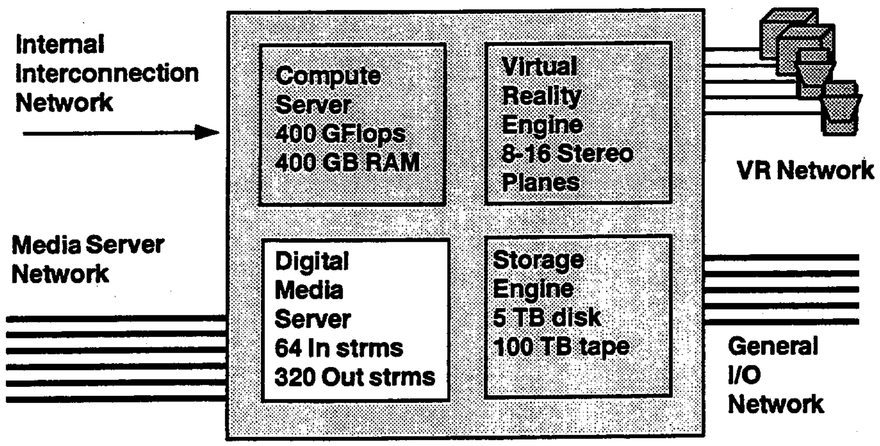
- Scalable interconnect of "rendering engines" to simulations
- Low latency network interface, WAN latency management
- Mechanisms for sharing models (simulation vs. rendering)
- Support for recording/playback
- Support of video data types
- New programming environments (open inventor++)



Current and Future Application Bandwidth Requirements

- Remote CAD prototyping (50 Mb/s)
- Typical 8-way LabSpace session w/archive (100 Mb/s)
- Cave to Cave applications (200 Mb/s)
- Nanotech design transmission (500 Mb/s)
- 8-way shared space CAVE session (1 Gb/s)
- APS detector data rate (1 Gb/s)
- Raw CAVE I/O (8 Gb/s)
- DOE Metalab backbone (500*100 Mb/s) ==> 5x10^4 Mb/s ==> 50 Gb/s
- Realtime earth system simulation (1 Tb/s)
- Distributed teraFLOP supercomputing (2 TF ==> 2 Tb/s)

Campus Digital Resource Engine


Internal Interconnection Network
Compute Server 400 GFlops 400 GB RAM
Virtual Reality Engine 8-16 Stereo Planes
VR Network
Media Server Network
Digital Media Server 64 In strms 320 Out strms
Storage Engine 5 TB disk 100 TB tape
General I/O Network

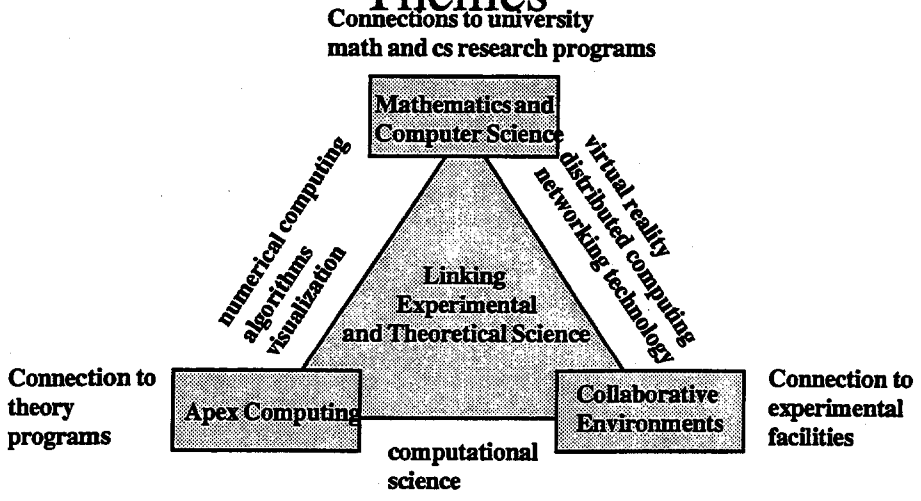
Core Digital Resource Engines

- Cycle Server
- I/O - mass store-- persistence for the environment
- Stream media -- page servers -- video/audio/web++/events
- Graphics -- geometry-rendering servers -- engines for visual immersion
- Networks -
  - internal component interconnects --> 1 GB/s/link
  - local fabric -- shallow networks --> 1 Gb/s/link
  - wide area -- deep networks --> 1 Gb/s/link

Software Environment

- object oriented --> multithreaded --> language independent mechanisms
- integration with communications and security
- integration with media stream support (multicast, real-time etc.)
- building blocks for collaborations
- brokering/proxy capable
- multi-paradigm
- storage integrated-- performance centric-- persistent
- dynamically downloadable
- peerable server/client/models

"Integrating Research Themes"


Connections to university math and cs research programs
Mathematics and Computer Science
numerical computing
algorithms
visualization
virtual reality
distributed computing
networking technology
Linking Experimental and Theoretical Science
Connection to theory programs
Apex Computing
Collaborative Environments
Connection to experimental facilities
computational science

Philosophical Goals Behind Voyager and Avalon

- **Techno-Empowerment**: recognizes the great disparity in productivity and cognitive skills present in groups and organizations. The key point is to amplify the **best and brightest**, the **problem solvers** and the **primary integrating thinkers.** In addition through technology, uplift the middle and margins and empower everyone to contribute to organizational goals.
- **Key problems**: scalable interactions and coordination.

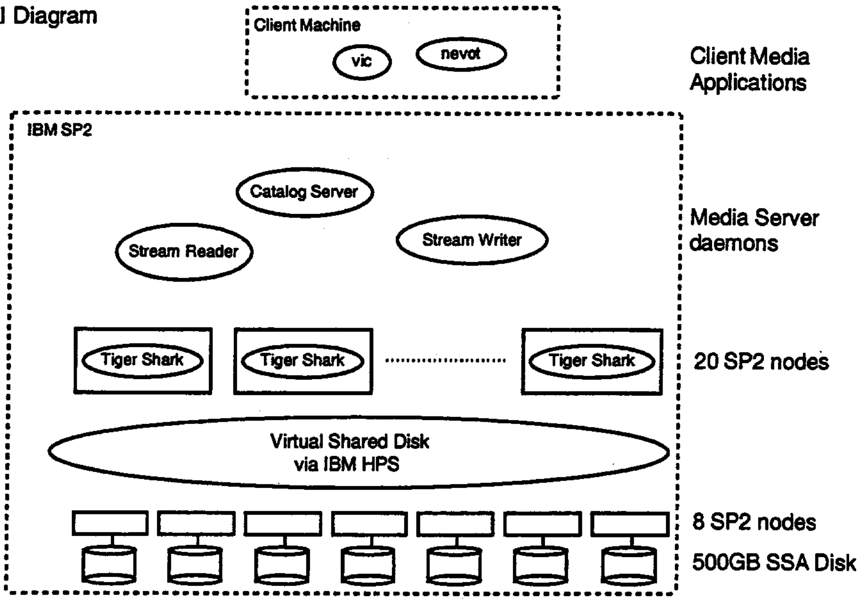
Classical Modes of Working

- Seminars (peers)
- Classes (students, teacher)
- Workshops (long duration)
- Meetings (short duration)
- Conferences (multisession)
- Panels/Roundtables
- Presentations
- Retreats
- Hotel Rooms
- Breakfast
- Collaborative Hacking
- Solo
- One-on-One
- Hallways-coffee breaks
- Help Desk
- Sales calls
- Walks
- Lectures
- Airplanes
- Lunch

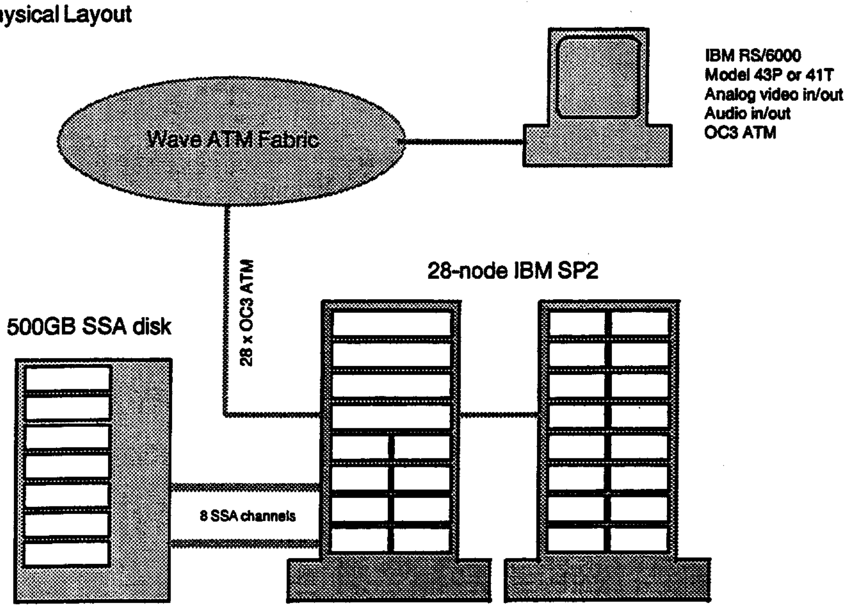
Argonne Voyager

- A project to develop servers to support the communications needs of virtual organizations.
- First effort is a video server (fully symmetric record, playback engine).
- Video server will be demonstrated at SC95 as part of the I-WAY activities.
- After SC95 we will be exploring support for a wide variety of data types and media streams.
- Scalability is a major goal.

SC95 ANL/IBM VideoServer
Logical Diagram



SC95 ANL/IBM Video Server
Physical Layout



Avalon: A One Thousand User Collaborative Environment

- Future research project at Argonne to put a thousand people on-line in a shared virtual environment.
- Prototype of an **Integrated National Collaboratory**.
- Scientific, support and administrative users.
- Based on **Voyager**, **LabSpace**, **Zipper** and **I-WAY** inspired technologies.
- Technology Goal: an ATM based 1000 desktop environment with interactive video, audio, text, web, computation, robotics and database resources.

Collaborative Environments -- Telepresence

- Provide the user with flexible ways to **"be there"**
- Includes both high-end tools (CAVE-to-CAVE) and desktop collaborative environments (LabSpace)
- I-WAY demonstration of telerobotic **"proxy"**
- Has major potential to change the way people work
  - telecommuting, telescience
  - remote control of experiments
  - distributed collaborative research groups
  - works for administrative as well as technical users
  - can reduce costs and reduce staffing levels

Distributed Collaborative Environments Technology

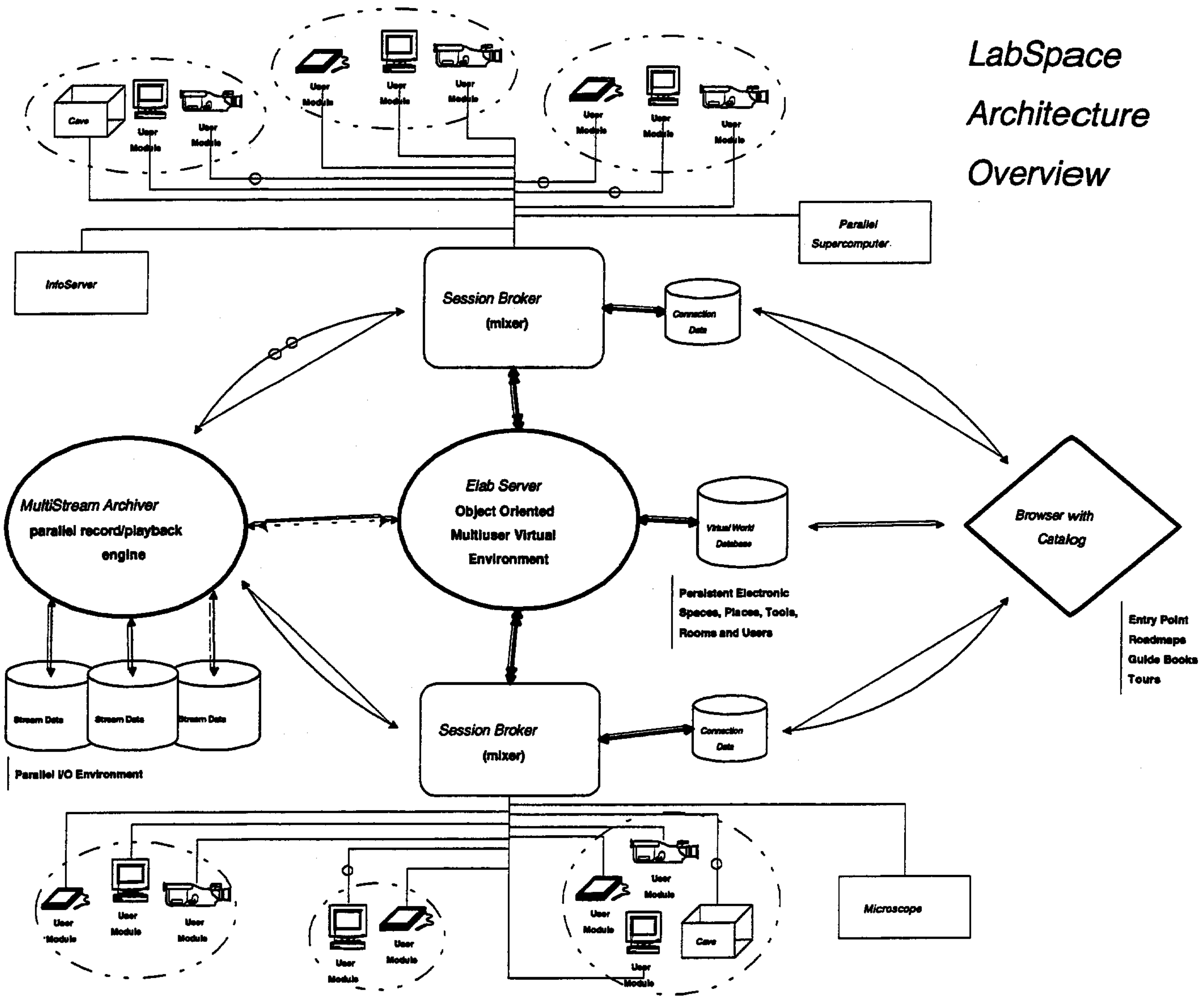
- Software and systems to support distributed scientific and engineering collaborations
- The **"LabSpace"** system combines audio, video and data streams into a **"telepresence'** environment to support remote operation and control of scientific experiments
- System will be piloted with users in Materials Science and CAT*s of the Advanced Photon Source
- Users will be able to collect and analyze data remotely and interact naturally with colleagues
- This technology can dramatically expand the utilization of DOE's ER facilities (e.g electron microscopes, light source beam lines)

* Collaborative Access Teams

Elab Server

- **Text based- multi-user virtual environment**
- Based on Lambda MOO server technology from Xerox PARC (e.g. BioMOO, AstroVR, InfoPark, WaterfallGlen, JHM, Lambda)
- Provides virtual environment "**context**"
- **Manages Database** of virtual locations, attributes, access mechanisms, users and objects
- Extensible and Object Oriented
- Scalable and portable

LabSpace Architecture Overview



FERMILAB-Conf-96/069

Using SNMP for creating Distributed Diagnostic Tools

O. Reisacher*, P. Ribeiro, H. P. Christiansen**

SL Controls Group

CERN, Geneva, Switzerland

*Left CERN in Feb. 1995

**Currently at UID, Linkoping, Sweden

Abstract

In this paper we describe how SNMP (Simple Network Management Protocol) can be extended to do control system diagnostics. Our solution consists of a SNMP agent for LynxOS and a configurable MIB (Management Information Base) browser. We have reused diagnostic modules from the existing diagnostic system and integrated our development into a commercial network management product.

Introduction

The control system for the LEP and SPS accelerators at CERN is a distributed computer system of over 300 nodes. This system must be operational during long periods of the year on a 24 hours a day basis. A number of diagnostic tools have been developed over the years to support the intervention of the on-call team, and they were also integrated into the Control Room operations environment. For the management of the large Local Area Network covering over 200 sq. km, a commercial product, HP OpenView is used. The project aimed at integrating these different tools into a common framework was started in 1993.

This paper is divided into five parts. We start with the description of the existing distributed diagnostic architecture and the motivations for this project. A brief description of the current network management standards follows. The description of an SNMP agent for LynxOS along with its enterprise MIB is followed by the presentation of a configurable MIB browser. The last part covers the integration of this framework into the HP OpenView environment.

1. Existing architecture and motivations for this project

The LEP and SPS control system is a distributed system that can be divided into three interconnected logical layers :

1. Back-Ends

At this level run the applications for the daily operation of the accelerators. These include not only all the application software related to normal machine operation, but also all the work related to anticipating control system failures and bringing the system back to a normal state after a breakdown. At this level, for system management, we have a home-made tool called xcluc that browses an information repository where all the periodic reports from the agents arrive. Xcluc presents the state of the different network nodes, enables access to the report information stored in the repository and also provides a number of primitives to interact with the different nodes. These systems are mainly X-Terminals running on HP 9000-7xx servers.

2. Front-Ends

The front-ends run a number of applications controlling the activity of a certain number of entities described in 3. These entities also act as proxies for the message-oriented acquisition and control scheme. A home-made diagnostic tool, clic (from the sound made when one takes a photograph) runs periodically acquiring a certain number of states pertaining to important sub-systems in this environment. These systems are PC or VME boards running the LynxOS operating system.

3. Equipment Control Assemblies (ECA)

These systems are doing I/O multiplexing acquisition and control. They are connected to the upper layer (the front-ends) by a message oriented protocol on top of a field bus. The status of the different ECA stations on the bus is periodically controlled by the clic application mentioned on 2.

The main motivation for this project was to develop an integrated system and network tool for operations and diagnostics. It was desired to reuse existing code from the clic application to but standardize the system information model and transport protocol aspects. As HP OpenView (OV) is used for the network management aspects, the basic idea was to transform the existing tools so that they could be easily interfaced to this package. Accordingly we took a number of decisions :

- use SNMPv1 with MIB II, supported by OV

- use the HP SNMP library

- develop a number of applications to support the integration

2. Standards

A number of Network Management standards have been developed over the past few years. These standards have a scope that goes beyond the area explored by this paper, system management. All the different architectures and standards for designing network management share the basic model that splits the Managing System and the Managed System into two separate sub-systems. Each of these sub-systems communicates with a Manager Kernel or an Agent and these communicate with each other in a client-server relationship via a Management Protocol. The applications, their environment and the system they run on are called a managing system. The network equipment and software is represented as a collection of managed objects. Those managed objects are supported by an agent. The managed objects, their agent and the computer system they run on are called the managed system. This management framework provides application programming interfaces (APIs) to the applications and objects. Of the three main management families, SNMP, CMIP (OSI) and DME (OSF), we have chosen SNMP to use as a framework for this project.

The management information model (SMI) is the most important part of a network management structure, as it defines the kinds of interactions between management applications and managed objects supported by the model. The managed objects are often active entities, their behavior being independent of any management system.

The SNMP family

The following refers to what is termed as SNMP version 1 as described in [RFC 1155, 1157, 1212]. There are three distinct sets in this family of standards :

- Management Information Base (MIB)

This specifies the variables maintained by the framework (this is the information that can be queried and set by the manager (MIB II, [RFC 1213])). This is in practice a text file describing the variables supported by a SNMP agent. For the sake of simplicity, a limited number of syntactical elements of the base syntax ASN.1 (Abstract Syntax Notation 1), are used. To avoid variations on the possible syntactical constructs defining the same object,a set of macros in ASN.1 notation is used [RFC 1212].

- Structure of Management Information (SMI)

Athis is a set of common structures and an identification scheme used to reference the variables in the MIB, described in [RFC 1155].

- The Management Protocol

This is a protocol layer on top of the User Datagram Protocol (UDP), implementing five basic types of transaction

SNMP GET

SNMP GETNEXT

SNMP SET

SNMP RESPONSE

SNMP TRAP

The philosophy behind SNMP is that the agent and the object implementation should be simple. In this context, the SNMP SMI only supports a limited set of interactions between the application and the object. Operations to Get and Set attributes (called MIB variables) are supported. Attributes can be gathered into tables and groups, but operations on tables or groups are not directly supported. Create, Delete and various Actions are all implemented as a side effect of setting a MIB variable. An individual MIB variable is named within an agent by an object identifier. Object identifiers are a sequence of integers, representing a path through a tree. Individual MIB variables in a system are represented by a leaf on this tree. The nodes above the leaf level represent tables, groups or MIB variable types. The GetNext operation is the only means to make an inventory, similar to a directory tree listing, of all the MIB variables supported by an agent.

3. SNMP agent for LynxOS and its enterprise MIB

The SNMP agents available for commercial systems are often closed to modification. Any extension capability that is available is often of limited scope. The first version of the SNMP agent for LynxOS was based on the Carnegie-Mellon University (CMU) source code.

Along with the extensible SNMP agent prototype, compliant to the MIB II specification, a ASN.1 parser has been developed. Each extension, new object, is described as a MIB object with the ASN.1 syntax and entered in the database. This description is run through the parser which generates the hooks to the C code to be called whenever an access to the new object is requested. The new code should be linked with the other SNMP agent modules to produce the extended agent (see figure 2). Using this method, the SNMP agent was first made to support the network management MIB. The private part of the MIB was then integrated in a stepwise fashion.

Figure 1 shows the main building blocks of the described SNMP agent. The system surveillance is performed every trap_interval seconds. If an error condition is detected a TRAP is sent to the manager. Trap_interval is a MIB object that can be set. Another agent module handles the SNMP requests originated by the manager. The extensions to the MIB describe the system variables that should be monitored and also the variables specifying the limits against which the system variables are tested. These variables are specific to the system to be managed.

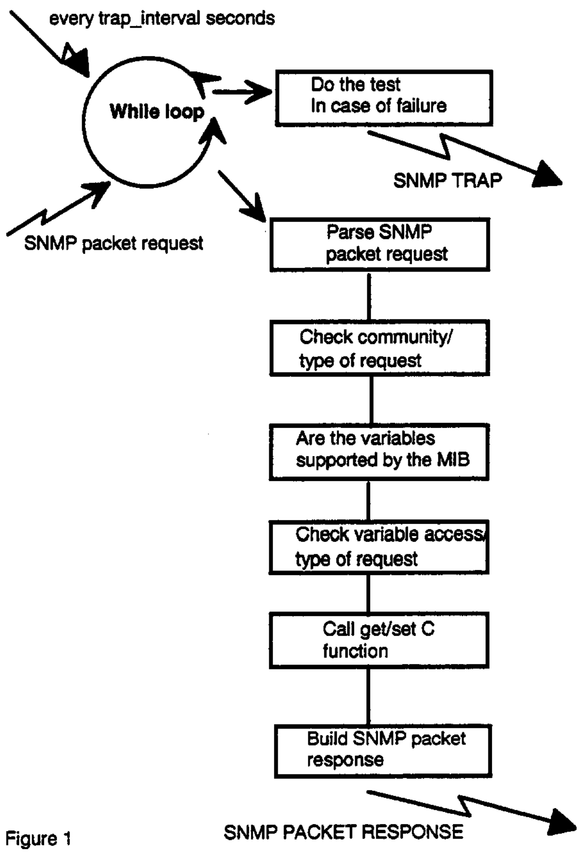

every trap_interval seconds
While loop
Do the test
In case of failure
SNMP TRAP
SNMP packet request
Parse SNMP packet request
Check community/ type of request
Are the variables supported by the MIB
Check variable access/ type of request
Call get/set C function
Build SNMP packet response
SNMP PACKET RESPONSE

Figure 1

The instrumentation of operating system parameters is not a trivial task. One of the main concerns of this project was to provide this information using a minimum of system resources. As our LynxOS systems run diskless, we had to implement and incorporate into the SNMP agent functionality similar to the one provided by system applications like **df**, **ps** and **netstat**. The agent is not **forking**, doing dynamic memory allocation or interacting with files once it is up and running. The reason is that this is the only way to be efficient when the node is in a bad state and needs diagnostic actions.

The following is an example of adding an object to the private MIB subtree. This example code is written for LynxOS, where many system parameters can be obtained using the **info** system call.

The ASN.1 definition for the MIB :

```
        max_inode OBJECT-TYPE
                SYNTAX INTEGER
                ACCESS read-only
                DESCRIPTION
                        "Read-only integer value corresponding to the system file table size"
        ::= { filesystem 1 }
```

The C function that implements the described object :

```
#include "snmp_impl.h"
#include "pdu.h" /* VarBind */
#include "error.h" /* ERROR_MALLOC, NO_ERROR */
int get_file_table_size ( var )
VarBind *var;
{
        var->value.integer = ( long *) malloc (sizeof (long));
        if (var->value.integer == NULL) return (ERROR_MALLOC);
        * (var->value.integer) = info (I_NFILES);
        var->value.len = 1;
        return (NO_ERROR);
}
```

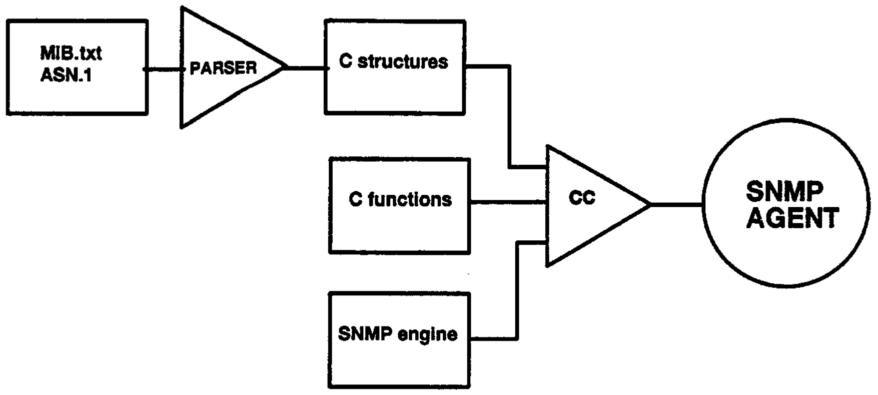


Figure 2

4. The development of a configurable MIB browser

The development of this product was decided at a late stage of the project. Apparently in contradiction with the project aim of integrating applications at the top level, this product was justified by the need for a simplified and user configurable view of a limited number of variables and tables.

The OpenView browser has many very powerful capabilities, such as the possibility to dynamically load/unload an MIB file. It also has a menu-based application builder, but it suffers from a number of drawbacks such as being too close to SNMP theory, producing an almost unreadable output of table variables and producing a complex set of windows,where the destination entity for the manager queries is not easily known all the time.

The new browser was developed using the Hewlett-Packard SNMP library and a Motif user interface.

Following the requirement specifications, the browser is capable of reproducing a session by saving the previous session layout in a configuration file, on a one per user basis. This enables a user to create a reproducible session profile by adding to the current session the objects pertaining to the required status view.

A clear distinction is made between columns in a table object and plain objects so that the presentation follows the object type.

All the SNMP related information is kept away from the user interface.

The browser can be configured to run in a single window mode.

The type of the data displayed can be changed with a “translator” function.

5. The integration of this framework into HP OpenView environment

The HP OpenView system allows the creation of submaps and different views. This has been exploited in this project. In addition to the view that IP Map offers, a secondary view, close to the existing repository browsing tool **xcluc**, has been created.

The usage of the IP Map application can continue as before. This view will reflect the combined network and system states of the nodes as reported by the extended SNMP agents.

With the API provided with HP OpenView, a simple interface to display parts of the extended MIB can be quickly built. The experience acquired by doing this will make extensions to similar interfaces concerning other objects in the MIB an even easier job.

The API also provides interfaces to the HP OpenView database to get hold of the received traps and also OSF/Motif functions that allow several applications to work with, and control, the visual aspects of a common submap hierarchy, thus cooperating with the IP Map application.

Conclusions

This project proved the feasibility of replacing the home-made distributed diagnostic tools by a SNMP-based framework. Among other benefits, we will be able to integrate the view of network and system diagnostics, use a standard protocol for the manager agent communications, provide a standard hierarchy for the managed data and reduce the maintenance effort to one agent.

We found some limitations and these included limited support at the data types level, very slow access time under SNMPv1 for table object retrieval, and that further studies were needed to be done on security and reliability issues.

References

Managing networks and systems with SNMP. O. Reisacher 1995, http://www.cern.ch/CERN/Divisions/SL/SpsLepControls/snmp_sysman.html

Installation and Management of the SPS and LEP control system computers, Alastair Bland, CERN, paper presented at ICALEPCS'93, Berlin, Germany

SNMP, SNMPv2 and CMIP, William Stallings, Addison Wesley 1993

FDL, a Deterministic 100 Mbytes/sec Data Link

J. Bovier, J-F. Gilot
Creative Electronic Systems

Abstract

Data transfers in real time applications needs more than just a speed increase to meet the new constraints of data acquisition and control systems. Data gathering from multiple sources, labelling, transfer scheduling and data scattering to multiple destinations are of primary importance.

Over the past few years the power of processing units used in real time data acquisition and control systems has increased by a factor of 100 (from 1 mips to 100 mips) thanks to the availability of RISC-based microcontrollers with clock rates well above 100 MHz. Instructions are now executed in less than 10 ns if cache memory is used. In order to make the best use of this power, external agents are needed to move data to and from processors' memories.

Intelligent I/O boards have been provided to relieve the processing units of direct instrument control and data transfer protocol management. They pack data in buffers ready to be grabbed for processing. The missing piece in this scheme is an autonomous system able to gather data from I/O boards and scatter it to processors' memories at speeds matching those of the processors.

The Fast Data Link (FDL) is a set of hardware and software tools designed to transfer data "intelligently" between multiple sources and destinations.

The backbone of the FDL is a multimaster/multidrop (up to 15 nodes) copper link synchronised at 50 MHz. The physical support is a cable composed of 25 twisted pairs (16 data - 2 parity - 7 control signals) extending over a maximum distance of 30 meters.

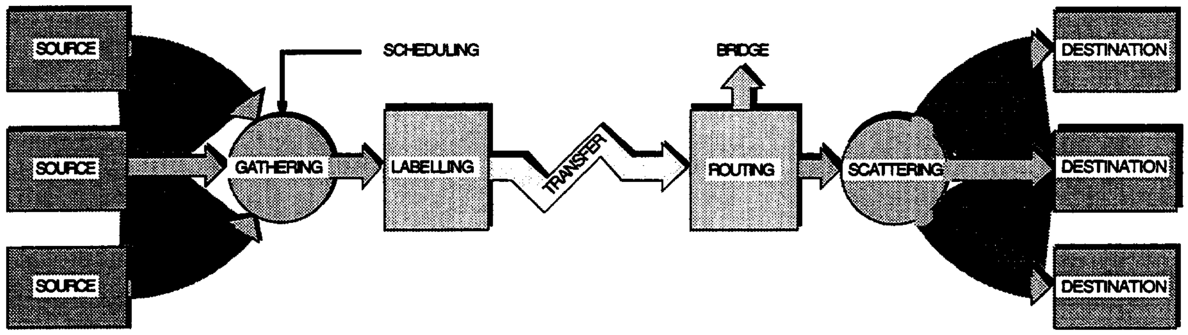


Figure 1. FDL General Concept

The client/server approach has been chosen in designing FDL interfaces. Upon request of a client or occurrence of an external event, the interface gathers data from data acquisition busses, dual ported memories or registers and stores it locally in an intermediate buffer. Information is then added to the data packet before transfer over the link so that each data packet contains routing parameters. In the destination nodes data packets are routed to their final destinations.

A point-to-point fibre optic link has been designed to interconnect at full speed local FDL networks over long distances. Up to 15 optical connections can be implemented from a local network.

Information is transmitted over the FDL in units of "cells". A cell is a sequence of 18 words (of 16 bits) emitted by the current master every 20 ns.

The link is source-synchronised. The current master drives the clock signal and sends one data word every clock cycle. The cell synchronisation signal indicates to the slave nodes the beginning of an 18 word cell. The first two words of every cell contain routing information (destination node identifier, cell identifier, destination buffer identifier, etc.) They are generated by the FDL master port and decoded by the FDL slave port. The payload of a cell is 32 bytes.

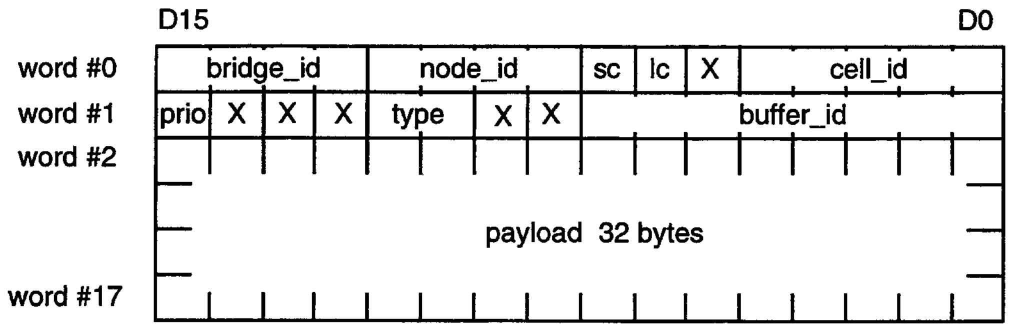


Figure 2. Cell Structure

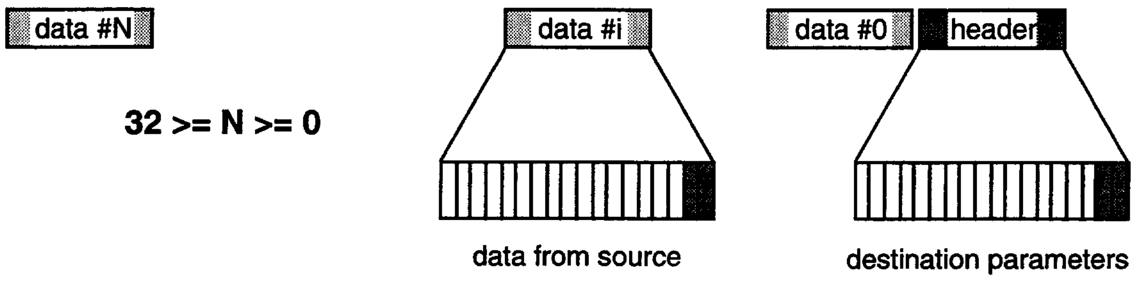


Figure 3. Data Packet Structure

For data transfer, cells are organised in packets of up to 1 kbyte (32 cells maximum) preceded by a header cell containing parameters for the final destination. Thus routing and administrative information imposes overhead of 14%.

Single cells are used for FDL management. The FDL offers two levels of priority for cell transmission to allow management cells to take precedence over data cells. When no node has cells to transmit, the current FDL master transmits empty cells to maintain link synchronisation.

The FDL is a deterministic link. The arbitration mechanism guarantees a defined data transfer rate and the maximum latency for bus mastership attribution.

To offer fair access to the link, bus arbitration is assumed by the current master. This implements a rotating arbitration. A requesting node asserts the Bus Request signal at the beginning of a cell; the current master grants the bus at the beginning of the next cell. This allows the next requesting node in the arbitration daisy chain to own link mastership before the end of the second cell following the Bus Request assertion. This mechanism fixes the latency for a change of master at 720 ns + time of flight between current master and requester (40-150 ns).

latency: < 1 µs

In addition the link shares ownership fairly amongst requesting nodes, thus guaranteeing minimum latency for data transfer completion. For a fully loaded link the overhead for arbitration is one time of flight every two cells. Depending on the physical length of the link the average arbitration overhead is then between 6 and 20%.

The nominal transfer rate of the link is 100 Mbyte/sec x (1.0-Overhead fraction). The average transfer rate is the nominal transfer rate divided by the average number of requesting nodes.

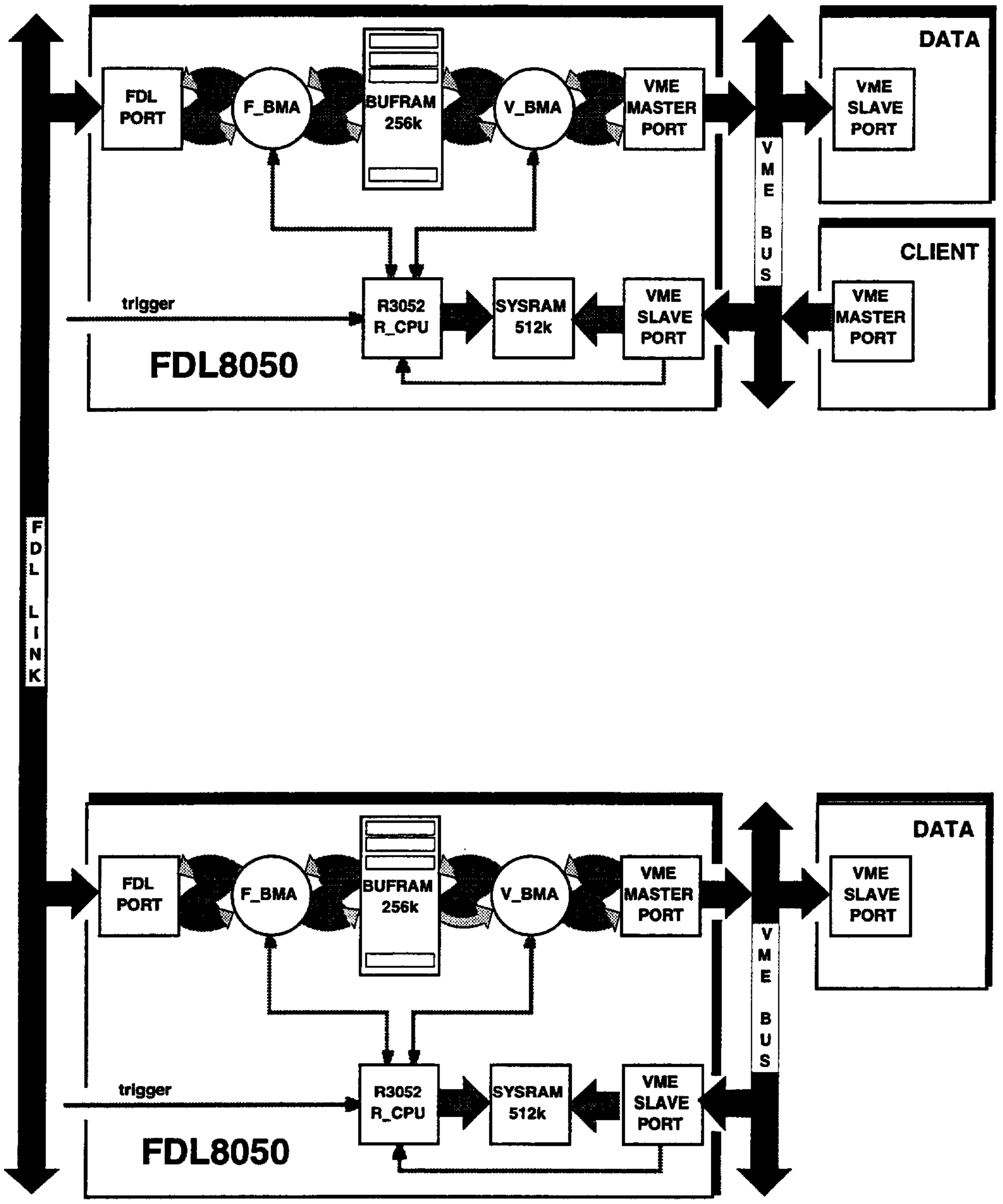


Figure 4. FDL8050 VME interface

The FDL8050 interface is a VME server designed to transfer data between VME crates at rates up to 70 Mbyte/sec. A client is any VME master able to access the FDL8050's VME slave port.

The central part of the VME/FDL interface is a fast dual ported memory (BUFRAM) accessed by two block movers at 200 Mbytes/sec. The first one (V_BMA) gathers/scatters data from/to the VME bus. It packs the data which it reads into blocks up to 1 kbyte in size. The second one moves data packets between the BUFRAM and the link. It handles link protocol and packet fragmentation into cells.

The two block movers are controlled by a RISC microprocessor (R_CPU (mips R3052)) managing data transfer and communication with clients. The system memory (SYSRAM) is accessible from both the V_BMA and the VME bus. This feature is used to implement a 64 kbyte mirror memory. External trigger lines are connected to the microprocessor. They are used for event generation.

The transmission mechanism uses request fifos (of 256 words) to keep both BMAs busy with data packets for transfer.

To trigger a V_BMA data transfer, the R_CPU prepares a transfer descriptor in the SYSRAM and then posts the descriptor index in the V_BMA request fifo. The V_BMA is built around an R3051 microprocessor in order to handle complex VME transfer descriptors. It checks for block boundaries and recovers from bus errors in less than 1 µs. At the end of the transfer the V_BMA updates the descriptor with the transfer status and writes the descriptor index into the V_BMA status fifo. If required, control information such as sequence number, time stamp, etc. is added to the data packet.

The R_CPU builds the header cell with final routing parameters in the HEADRAM, and posts the cell index in the F_BMA request fifo. The F_BMA then transfers the header cell followed by the associated data cells over the link. At the end of the transfer the cell index is written in the F_BMA status fifo.

The destination node catches cells containing its own node identifier and stores them in the BUFRAM. The HEADRAM holds the header cell and the F_BMA status fifo the cell index. The R_CPU reads the header cell and builds a transfer descriptor for the V_BMA. It then triggers the V_BMA to transfer the data packet from BUFRAM to VME. A single cell containing an acknowledge from the data packet transfer is sent back to the transmitter for flow control.

A 12-stage transmission pipeline has been implemented to maximise data throughput while controlling arrival of data packets. Up to 8 VME transfers can be concurrently active.

High priority single cells (service cells) are transmitted over the link for system management; they take precedence over data cells for link access A dedicated F_BMA request fifo is used to post these cells, the HEADRAM contains room for 256 of them, which are directly handled by the R_CPU. These service cells are used for data transfer acknowledge, remote request, event generation and time synchronisation.

The FDL supports time stamping of data packets and transfer synchronisation (isochronous mode). To this end every FDL interface has a hardware clock running at 1 MHz. The interface maintains a calendar time accessible by VME clients. One node in an FDL system keeps the global time reference and every 10 ms it broadcasts this reference value to other nodes. Time synchronisation service cells are directly handled by the interface hardware.

Special timers (25 MHz) on FDL interfaces allow a time synchronisation of all nodes with an accuracy of 1 µs.

The FDL supports two scheduling modes for data transfers. In the asynchronous mode they are triggered by an external event (client request, VME interrupt, front panel trigger, etc.) and are handled by the server on a first-in first-out basis. In the isochronous mode data transfers are activated periodically at a user-chosen frequency. The two modes of data transfers can be mixed

When the isochronous phase occurs, all asynchronous data transfers are suspended and a pre-defined transfer descriptor is given to the R_CPU for processing. At the end of the isochronous phase (the duration is programmable) asynchronous data transfers are resumed A special F_BMA request fifo is used to post isochronous data transfers in order to allow instant switching between isochronous and asynchronous modes.

The isochronous mode has been implemented to allow all FDL interfaces to activate special data transfers periodically at the same time. This ensures the time coherence of the data collected by the system.

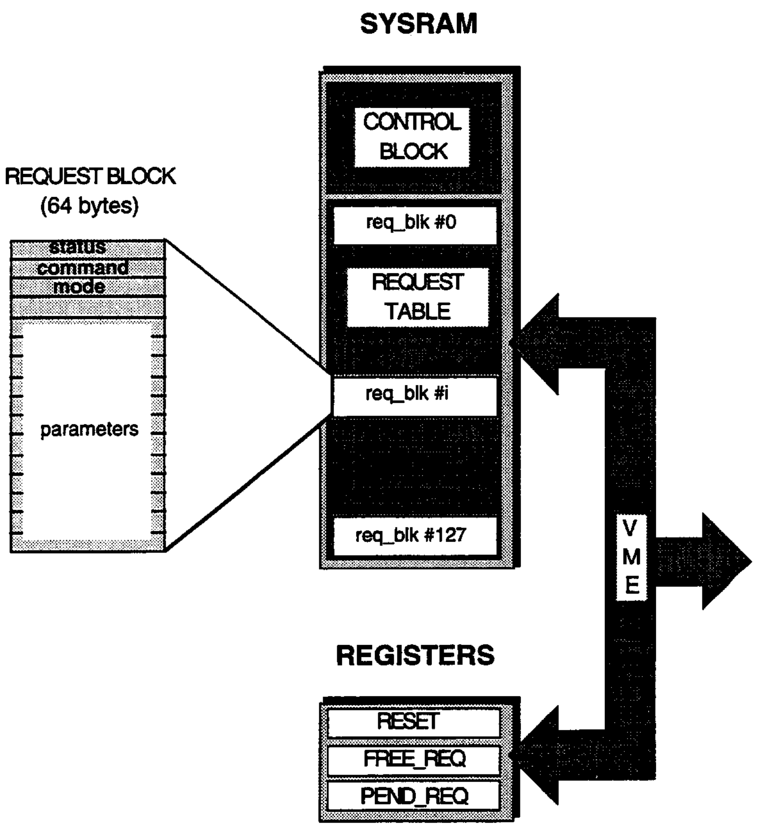


Figure 5. FDL8050 programmer's interface

The VME clients interact with the FDL interface through a shared data structure located in the SYSRAM. Two hardware registers (FREE_REQ and PEND_REQ) are used to control access to that structure. Remote reset of the interface is activated by writing in the reset register. Up to 127 connection channels can be opened at a time to these VME clients. Each communication channel is controlled by a 64 byte request block in which the client posts the transfer parameters.

The FDL firmware supports direct and indirect data transfers. For direct data transfers the client provides all routing parameters in the request block. In the indirect mode, the client provides the reference of a transfer descriptor. Transfer descriptors are linked lists of elementary transfer parameters (chained list of buffers, direct rings, indirect rings, etc.). These transfer descriptors can be pre-defined through a resource creation mechanism, or can be read by the FDL at transfer time. The FDL firmware supports the creation of remote transfer descriptors in order to handle indirect transfers in destinations.

The server supports 8 levels of priority for client requests and handles programmable time-out. Broadcast to multiple VME destinations has been emulated in order to implement protocols such as TCP-IP.

Device drivers for the most popular operating systems have been written to control the FDL8050 interface. If the driver is linked with the TCP-IP layer, a workstation can be used as a gateway to pass TCP-IP packets over the FDL. This allows a VME multiprocessor system to be controlled from a local area network while fast data transfers are going on between VME crates.

Interfaces to other busses such PCI, CAMAC, FASTBUS, etc. are under development in order to build complete data acquisition and control systems. We feel that PCI will be a standard in future workstations. The PCI/FDL interface will be used to push data directly into workstation memories under control of the operating system, so the data will be directly available to data processing software.

We will also provide embedded FDL interfaces to be directly implemented in instruments and detectors in which a register or a dual-ported memory will be used to post data for transfer over the link.

The performance of a real time system depends mainly on three parameters: the processing power, the interrupt response and the data transfer rate. The only way to maximise all these parameters simultaneously is to make them independent by dedicating specialised hardware to each of them. The FDL has been designed to maximise data transfer without compromising processing power or interrupt response.

Using World-Wide-Web for Control Systems

F.Momal, C. Pinto-Pereira
AT Division CERN, 1211 Geneva 23

ABSTRACT

World-Wide-Web is really welcome in control systems. From anywhere, operators and specialists may access data from the control systems without having to install specific software or without bothering about the type of computer to use. Nearly any computer, loaded with a little piece of world-wide available software, may have access to the archives files of the controlled process or its documentation. The Web standard, based on the hypertext principles and being highly focused on the man-machine interface, is an **easy and intuitive way** of accessing data.

The paper describes the link which has been made between the WWW and the LHC magnet test bench control system. It lists the different kinds of relevant data which are presented through this network tool: user-guides, on-line help, process documentation and archived data, but also **real-time data** like trends of sensor values or alarms. A software **gateway,** decoupling the Web server and the control system is described.

A structure has been designed to cope with the nuisance which could be brought by the Web, as well as to handle security features. It offers full transparent access to all the different data wherever they come from: files on several computers, databases, control system's real-time values, etc.

INTRODUCTION

A test bench has been built at CERN to accept and measure the numerous superconducting magnets for the future LHC. CERN has also built and commissioned the first version of the LHC Test String which is composed of several magnets in line. The control of the test bench[3] and of the string[4] is carried out by means of industrial components[1,2]. The process control uses standards PLCs connected by an ethernet plant network. Supervisory software systems carry out the man-process interface and some of the data collection. On the string, a fast data acquisition system continuously stores the status and values of a set of sensors and transfers them to an Oracle Database.

To offer remote access to the process and to allow easy post-mortem analysis, we have installed a World-Wide-Web server and developed a gateway between this server and the control system components.

THE ADVANTAGES OF THE WWW

As soon as the tests started, remote access was immediately needed to the control data and to the archives. For this reason, we developed a number of tools under MS Windows. Some of these provided real-time access to the control system, some allowed extraction of sensor values stored in the archives and others provided the states of the alarms and of the control software. The user-guides were made available on a Novell server. Various protocols were used (telnet, TCP/IP sockets, rsh, MS Windows DDE, etc.) and numerous developments were needed. The overall complexity, both for the developers and for the users, grew rapidly. The maintenance of these products became time consuming. In addition, only PC users could get at this information. For these reasons we became interested in the WWW[5], which has the following advantages:

For the developers:
- No development needs to be done on the users' machine.
- The Web software is obtainable in the public domain and needs no extra maintenance. New versions are released frequently.

- Browsers are available on most common platforms and are system independent, the only requirement being a connection to Internet and a standard WWW browser such as Netscape[8]. Even relatively "out of date" machines may be used (PC 386 with 4Mb of RAM).
- The multimedia capabilities of the browser enable the data to be presented not only in text but also in graphical form.

For the user:
- The interface is standard, graphic and intuitive. The learning time is minimum.
- A unique interface replaces all the previous small packages.
- The user does not have to worry about the source of the information he wants to see. Transparency is maintained by the server which may transfer, in the same way, to the user a document stored on a PC or a list of values coming from a centralized database.

However, a certain number of restrictions need to be taken into account:
- We should limit the transfer data format to the built-in capabilities of the browsers. This include ASCII or HyperText Markup Language (HTML)[6] for the text and JPEG or Graphics Interchange Format (GIF)[7] for the graphics.
- Transactions between clients and servers are atomic. The communication links close down after each transfer of a document. The user has to press the reload button to update the information (however a non-standard extension to the Netscape browser allows to do this automatically).

We decided to install a WWW server to meet the users' requirements for remote access. A home-made gateway fills the gap between the WWW dedicated to static documents and the control system, which delivers data which change in real time.

THE USERS' NEEDS

The server satisfies several needs:
- Those of the **process experts**: they want to follow the process behavior in real-time, to be informed of the evolution of a certain sensor, to correlate data coming from various programmable logic controllers (PLC), etc.
- Those of the **physicists or the managers**: they need more specific or general views of the controlled process, they want to retrieve and reprocess various data from archives, etc.
- Those of the **control system experts**: they want to monitor the behavior of their system.

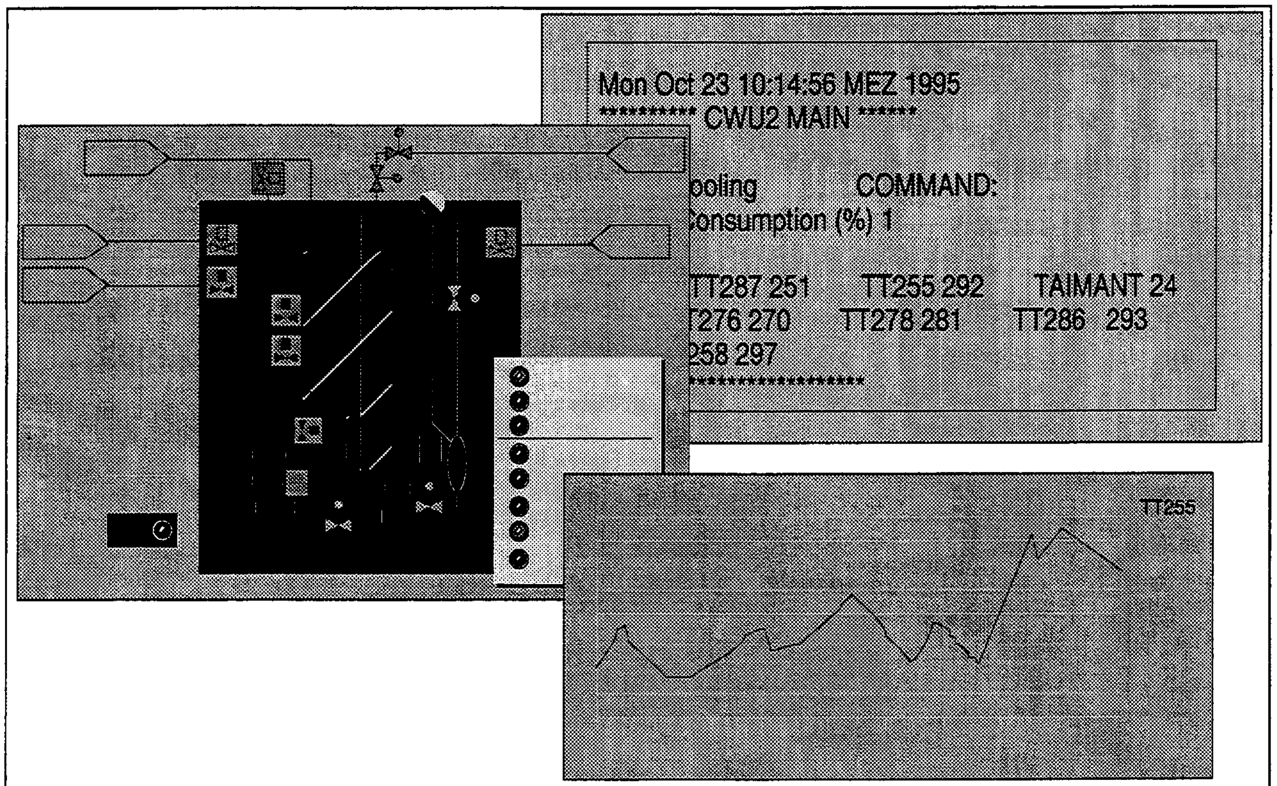
The goal here was not to make a new control system, but to built a complementary system. It allows the user to have a remote view of the data which is more restrictive but more general. Archive analysis tools were added.

DATA AVAILABLE ON THE SERVER

To answer the previous needs, we have made available several types of data:

Real-time process data
These are the data of the ongoing process. They are normally located in the control machines which are directly acquiring sensor data and/or are doing the supervision of the process. They are presented in the following formats (fig. 1):
- Graphical synoptics: they are generated on the fly and offer a process view which is similar to those presented on the supervision consoles. To offer a faster understanding of the process status, the views may be simplified by presenting only the most relevant data.
- ASCII tables: tables of real-time data may be retrieved. Functionally similar to the synoptics but without the graphical part, they are faster to retrieve and may be stored on the client machine for further processing. These tables can be customized at will.
- Graphical trend curves: the recent trend of the variables may be presented graphically.
- List of values such as the alarms of the process.

Archived data
These are all the data acquired by the supervision system and by the fast acquisition system. Supervision consoles store continuously the values coming up from the process in flat files, while the fast data acquisition system stores the acquired sensor data in an Oracle database. In both case the user may request, using an HTML form, the list of values taken by a specific parameter between two times. The result is sent back either graphically or in an ASCII list which can be reprocessed on the client machine.


```
Mon Oct 23 10:14:56 MEZ 1995
********* CWU2 MAIN ******

ooling          COMMAND:
onsumption (%) 1

TT287 251     TT255 292     TAIMANT 24
276 270     TT278 281     TT286  293
258 297
****************
```

Figure 1: Different types of real-time data available through the WWW server

Status data of the control system
Pages generated on the fly give the status of the supervision software and of the connections with the PLCs.

Documentation
The documentation linked to the control of the tests is available on line (User guides, On-line Help, process documentation, information on the people in charge, etc.).

HARDWARE AND SOFTWARE IMPLEMENTATION

Hardware structure
Two machines are dedicated to the WEB server:
- A Pentium under MS Windows which stores the static documents such as the user-guides. The Windows httpd WWW server is used[9].
- An HP 9000/715 station running HP-UX A.09.05 which uses the CERN httpd server[10]. The home page of the server is located on this machine which is essentially oriented toward the interconnection with the control system.

With such a division on our server, we limit the load on the workstation and thus we can respond faster to the real-time data requests. The Uniform Resource Locators (URL) of the documents stored on the PC are referenced on the main pages of the HP so that the user is not even aware of this separation.

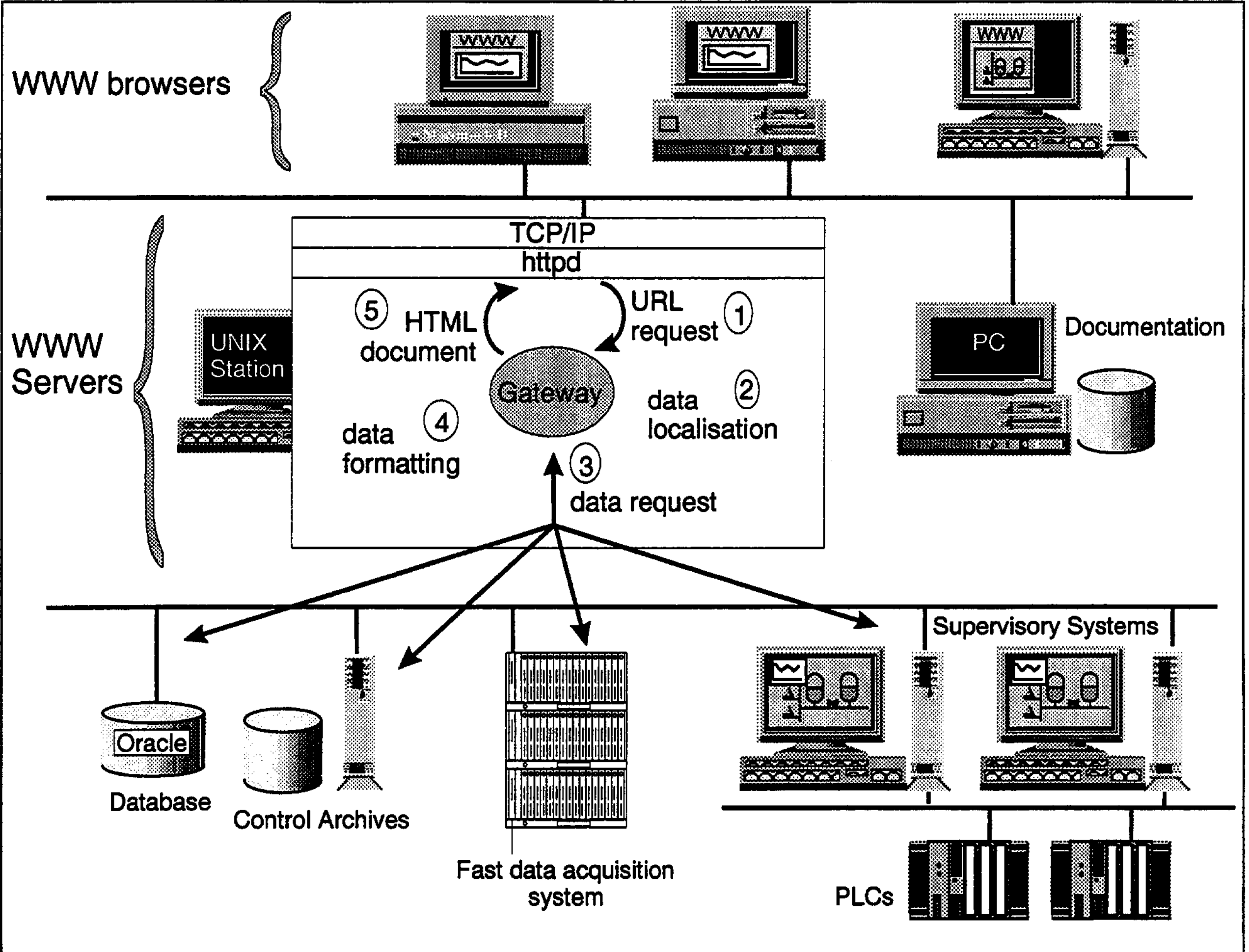


Figure 2: Gateway Implementation

Dynamic data handling

The CERN WWW httpd server software, like the others, offers the possibility to execute an external program. These programs are called gateways and use the Common Gateway Interface[11]. The gateways are linked to an URL and are executed whenever this URL is requested. Parameters may be sent with the URL. We have used this functionality to generate documents on the fly containing the requested process data (supervision and archives data). These documents are formatted using the HTML standard before they are sent back to the user.

The gateways we have developed have two purposes (fig. 2):

1. The retrieval of the data needed by the request. A description as to where the data can be found is located on the server. The gateway uses this description to retrieve the data from the corresponding hosts. Depending on the data, different protocols are used (SQL*Net, nsh, rsh, etc.). A communication is established with the supervision system whenever real-time data are required. To minimize the load on the supervision system, only numeric values are retrieved.
2. The data formatting. As described previously, the data may be presented in several formats, some being graphical (synoptics and trend curves). In such a case, graphics have to be created in a WWW compatible format. We have chosen the GIF format. Generic programs have been written in C to dynamically create

synoptics and curves in this format (the graphical library Gd[12] has been used). Using a description on how the data have to be presented, the gateways create the HTML pages and the included graphics if needed. The result is then sent back to the user. Some additional information such as a time-stamp is appended to the result. This is particularly important in the WWW environment where transactions are atomic.

As we have already mentioned, there is no way under the WWW standards to automatically update real-time data. Nevertheless we used the Netscape extension which allows the user to request periodically a page. By that means the information displayed on the Netscape browser is continuously refreshed. We decided to use this extension because it does not cause any compatibility problems. This feature simply does not work on the other browsers.

SECURITY AND CONFIDENTIALITY

Security and confidentiality are sensitive points which have to be looked at on a Web server. The very nature of the system and its location on Internet makes it more accessed. It is not sufficient to keep private an address. Before we even made public our server, we counted 210 visits coming from numerous countries (including France, Austria, United Kingdom, Belgium, USA, Australia, Italy, Canada, Holland, Japan, Sweden). Automatic search engines also entered the system. Log files allow us to trace all the accesses.

While most servers seek popularity, this is discouraged in this particular environment. In particular:

- The system accesses raw data which is not intended for general broadcast since they may be misleading before interpretation. For example a badly calibrated sensor will send apparently false data.
- We don't want to attract unnecessary access to the control system even if the data flow is regulated.
- Formatting data graphically is time consuming and the normal user would be penalized by unwanted visitors.

We must nevertheless take into account the possibility of an ill-motivated intervention. This risk would be further increased if writing should be possible through the server. Because reading of directories may be possible, they have to be protected.

Two levels of protection are used on our Web server. The first one filters the calling address. For some pages and applications, entry is allowed only if the user is at CERN. A second level, which is more restrictive, requires a password for entry. This is mostly done when access to real-time data is required. No illegal access was noted since we have implemented these protections.

CONCLUSION: ACQUIRED EXPERIENCE

The fast increase in the usage of our server confirms the choice we have made of the WWW as a means to access data coming from a control system (the number of accesses[1] went from 3.000 in March to more than 20.000 in September, with an estimated number of more than 30.000 in October). We can now present in a friendly way numerous types of data coming from different sources. Graphics created on the fly give the user a convenient view of the system. The CERN httpd server software on the HP has been now running for a year, so we have had the opportunity to test its robustness and reliability. Not a single problem was noticed since the beginning, which is not the case for the Windows httpd server which crashed four times in the past six months. Since we implemented the protective filtering system, no illegal access has been noted. The browsers that we use (Netscape and Mosaic[13]) are highly compatible and evolve to even greater levels of reliability and user-friendliness.

REFERENCES

[1] F.Momal, J.Brahy, R.Saban, P. Sollander, "Integrating a Commercial Industrial Control System to the Accelerator Control System", Proc. ICALEPCS 1993 Berlin, p 464

[1] An access corresponds to the acquisition of the document whether it is an HTML page, an icon or a graphical synoptic.

[2] R.Saban, P.Ciriani, A.Guiard-Marigny, H.Laeger, M.Rabany, A.Swift, "Equipment Industrially Controlled", Proc. ICALEPCS 1993 Berlin, p 461
[3] F.Momal, D.Bienvenu, D.Brahy, D.Lavielle, R.Saban, B. Vuillerme, L.Walckiers, "A Control System based on Industrial Components for Measuring and Testing the Prototype Magnets for LHC", Proc. EPAC 94 London, p. 2322
[4] R.Saban, D.Brahy, J.Casas-Cubillos, D.Lavielle, L.Madaro, A. Rijllart, M.Skiadelli, "The Control and Data Acquisition of the LHC Test String", Proc. ICALEPCS 1995
[5] The World Wide Web Consortium, http://www.w3.org/hypertext/WWW/
[6] HyperText Markup Language (HTML), http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html
[7] The Graphics File Format Page, http://www.dcs.ed.ac.uk/~mxr/gfx/
[8] Netscape's Home Page, http://home.netscape.com/
[9] Windows httpd, http://www.city.net/win-httpd/
[10] CERN httpd, http://www.w3.org/hypertext/WWW/Daemon/
[11] CERN Server CGI/1.1 Script Support,
http://www.w3.org/hypertext/WWW/Daemon/User/CGI/Overview.html
[12] A graphics library for fast GIF creation , http://siva.cshl.org/gd/gd.html
[13] NCSA Mosaic Home Page, http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html

The APS Control System Network

Kenneth V. Sidorowicz and William P. McDowell

Argonne National Laboratory

Abstract

The APS accelerator control system is a distributed system consisting of operator interfaces, a network and computer-controlled interfaces to hardware. This implementation of a control system has come to be called the "Standard Model." The operator interface is a UNIX-based workstation with an X-windows graphical user interface. The workstation may be located at any point on the facility network and maintain full functionality. The function of the network is to provide a generalized communication path between the host computers, operator workstations, input/output crates and other hardware that comprise the control system. The crate or input/output controller (IOC) provides direct control and input/output interfaces for each accelerator subsystem. The network is an integral part of all modern control systems and network performance will determine many characteristics of a control system. This paper describes the overall APS network and examines the APS control system network in detail. Metrics are provided on the performance of the system under various conditions.

INTRODUCTION

Figure 1 gives an overview of the complete APS computer network including that portion of the network which is used to control the accelerators. All of the APS networks, including the accelerator control network, use optical fiber

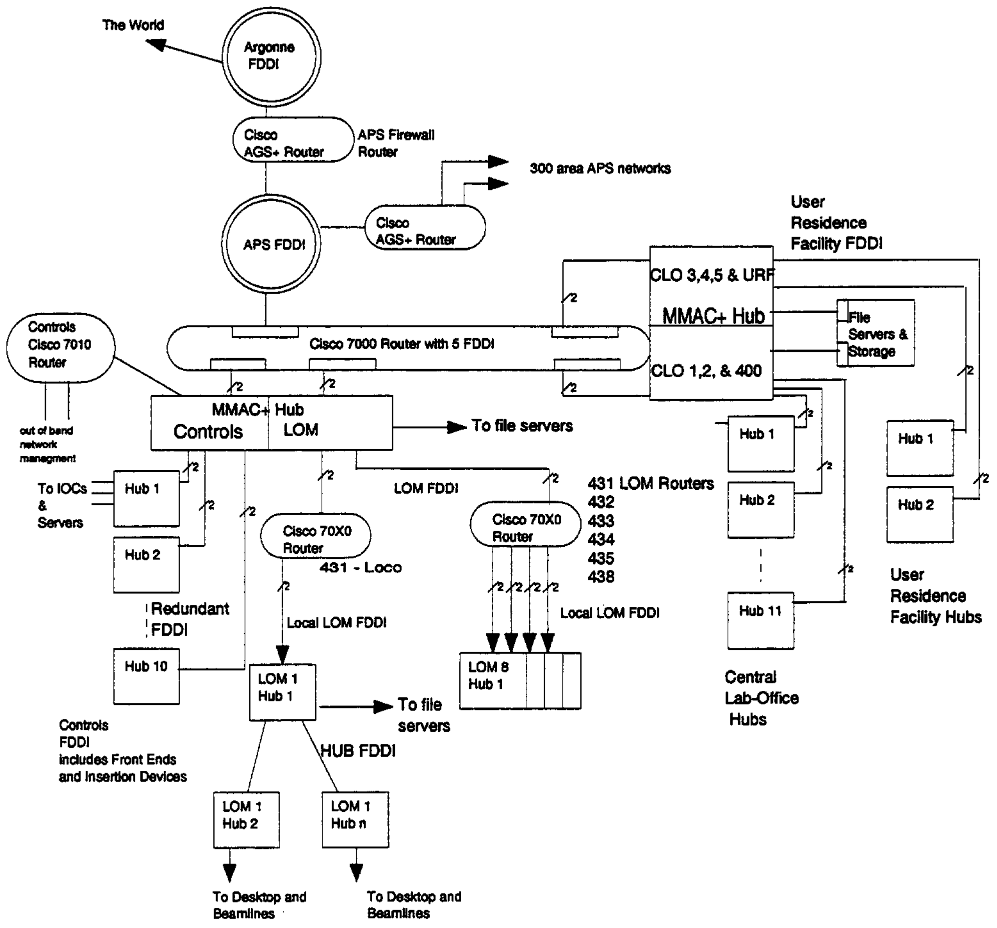


Figure 1. The APS Network

to connect satellite network hubs to a collapsed-backbone FDDI concentrator system. All the hubs are dual attached to the concentrator using a star connection configuration. The APS FDDI connects to the Argonne central FDDI network and then to the internet through a firewall router. The APS FDDI ring is presently configured to serve two locations 1.5km apart. In a few months the APS staff members will be located at one point and this FDDI ring will function only to implement the firewall and provide a defined point where network responsibilities shift from the APS network staff to the Laboratory network staff. The APS network must serve several diverse functions including accelerator control, beamline control, experimental data acquisition and normal day-to-day computerized office functions such as word processing. A router is used to isolate the APS network functions. The network lis divided into the control system, the beamlines in the Laboratory Office Modules (LOMs), the Central Laboratory Office Building (CLO), and the User Residence Facility (URF).

Each LOM is provided with a secure network closet in which all data network wiring is terminated and which houses the interface equipment needed to connect the collaborative access team (CAT) offices, beamlines, and CAT-provided local computers to the APS-provided network equipment. As shown on Figure 1, each LOM includes a router and a hub system which provides slots for Ethernet and Category 5 interfaces to the labs, offices, and beamlines. As they are needed, additional and higher-performance cards can be added to provide FDDI/CDDI (100 Mb/s) or ATM (up to 600 Mb/s) service to the CATs. Routes directly over planned national ATM networks will allow CATs to route data or "virtual presence" control between their home institution and the APS.

The network hub equipment and cabling plant allows an upgrade path to ATM technology (600 Mb/s) when this is needed. In addition, all connections and equipment will allow fail-over to redundant paths and equipment.

The same equipment and strategy has been followed in the CLO and the URF buildings. Thus, as with the LOMs, the computers installed in offices, labs, and residence rooms will use Category 5 wiring at 10 Mb/s Ethernet rates and be able to upgrade to 100 Mb/s CDDI rates on an incremental basis.

THE ACCELERATOR CONTROL STSTEM

The APS accelerator control system has been implemented using the Experimental Physics and Industrial Control System (EPICS) software tool kit. EPICS supports the "standard model" distributed control system which consists of operator interfaces, a network and software-controlled interfaces to hardware. At the APS, the operator interface is a UNIX workstation with an X-windows graphical user interface which may be located at any point on the accelerator control network and maintain full functionality. An operator has the ability to generate and alter control displays and to access applications such as the alarm handler, the archiver, interactive control programs, custom code and other tools. There are 16 operator interfaces directly connected to the APS control system.The crate or input/output controller (IOC) provides direct control and input/output interfaces for each accelerator subsystem. The standard crate uses either the VME or VXI standard, a Motorola 68040 processor, network communications and a variety of signal and sub-network interfaces. The 68040 processor provides the crate with the intelligence to allow it to run its software autonomously with respect to all other devices in the system. The software running in the crate hides hardware dependencies from the high-level software running on the workstation. There are approximately 130 crates used in the accelerator control system. A real-time operating system, VxWorks, is run in the crate CPU to provide the basis for the real-time control.

EPICS uses the TCP/IP networking protocol, a commercial standard supported by all network hardware vendors. The TCP/IP implementation is independent of the particular network medium selected to implement the network.

The function of the APS controls network is to provide a generalized communication path between the host computers, operator workstations, input/output crates, and other hardware that comprise the control system. The network design has taken into account the networking requirements of the EPICS networking interface, Channel Access (CA).

Channel Access is a 'software bus' which allows various EPICS entities to communicate over the network. Channel connections are made when a CA client starts up and sends a network broadcast containing the unique name or list of unique names of the process variables that it wishes to find. Each channel access server on the subnet searches it's list of process variables and establishes a TCP connection with the client if any desired process variables are found.

This behavior dictates the overall design of the control system network. It must be internally bridged in order to support the CA broadcasts and it must be separated from the other APS networks by a router to provide security for the accelerator controls.

A block diagram of the APS control system hub network is shown in Figure 2. The APS router, a Cisco 7513, is used to provide subnet separation for the control system and to provide the controls network with a subnet address.

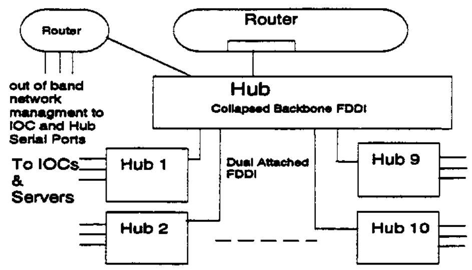


Figure 2. APS Control System Network

The router has a 100MHz MIPS 4600 processor and a 2-Gb/s backplane. An FDDI output card in the router is attached to one of the collapsed backbone FDDI buses on the Cabletron MMAC Plus enterprise hub. This hub was selected to provide high availability network services to the APS control system. The MMAC Plus hub accommodates 14 interface modules and has fault tolerant features built into it. It has two dual FDDI networks which provide up to 400 Mb/s of network bandwidth and it will support both packet and ATM cell transport.

There are ten satellite Cabletron MMAC hubs in the controls network which are dual attached to the central MMAC Plus hub using FDDI. The MMAC hubs are distributed throughout the accelerator facility to provide local Ethernet connections to all network devices. The aggregate throughput of the device is 26,180 packets per second. There are four hubs serving the storage ring, two serving the rf system, two serving the injector system and two serving the systems in the main control room. All the hubs are connected to the MMAC Plus hub using a star configuration. This allows us to reconfigure the network if future technology, such as ATM, should be installed. All of the control system IOCs are connected to the hubs using fiber Ethernet and thus they can be reconfigured if required. To provide redundant service, every IOC is connected to two hubs using a redundant fiber Ethernet transceiver: a primary hub with fibers running clockwise from the IOC to the hub and a secondary hub with fibers running counterclockwise from the IOC to the hub. This allows a hub to be serviced or to fail without causing the IOC to lose communication with the network.

Figure 3 shows a typical connection of the vacuum/power supply IOCs in five sectors of the storage ring. The IOCs are grouped in 20 locations around the storage ring. There are additional IOCs not shown in this area which are used to control the diagnostic devices, the front-end equipment and the insertion devices and to run the global orbit correction system. These IOCs are connected in the same way to the vacuum/power supply IOCs but are omitted here to simplify the diagram.

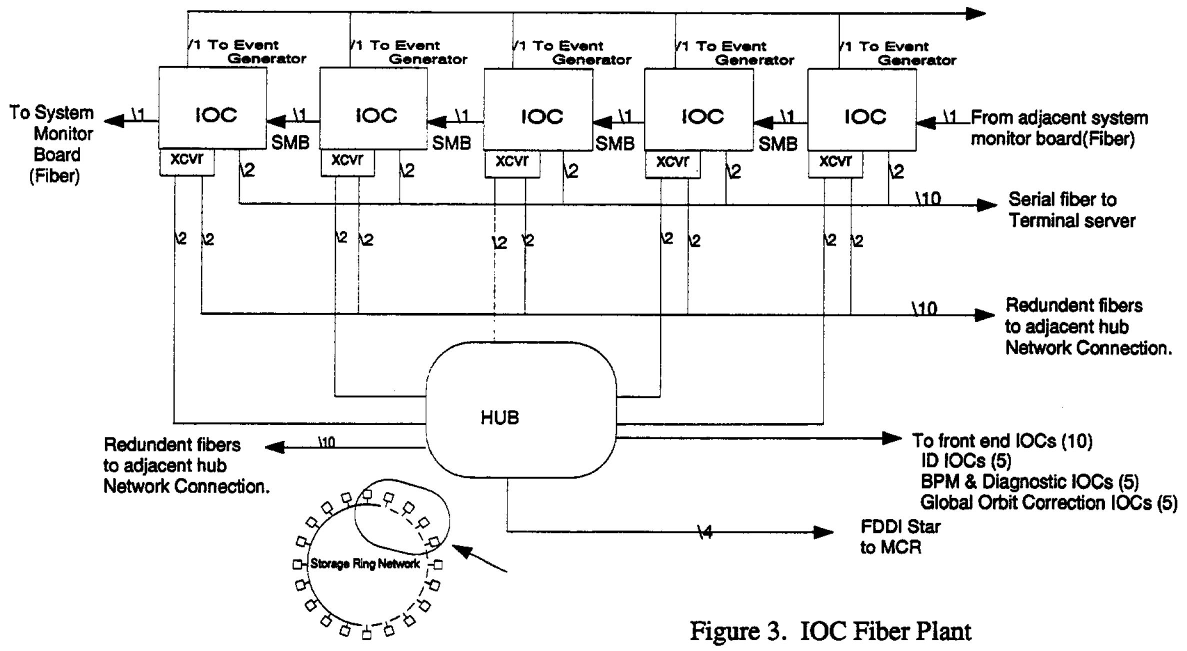

1 To Event Generator
IOC
SMB
xcvr
To System Monitor Board (Fiber)
From adjacent system monitor board(Fiber)
Serial fiber to Terminal server
Redundent fibers to adjacent hub Network Connection.
HUB
Redundent fibers to adjacent hub Network Connection.
To front end IOCs (10)
ID IOCs (5)
BPM & Diagnostic IOCs (5)
Global Orbit Correction IOCs (5)
FDDI Star to MCR
Storage Ring Network

Figure 3. IOC Fiber Plant

NETWORK TESTING

Normal network traffic on the control system FDDI is shown in Figure 4. With all accelerator systems running, the average network utilization is close to 4%. On October 13th the storage ring was not operating. Beginning at

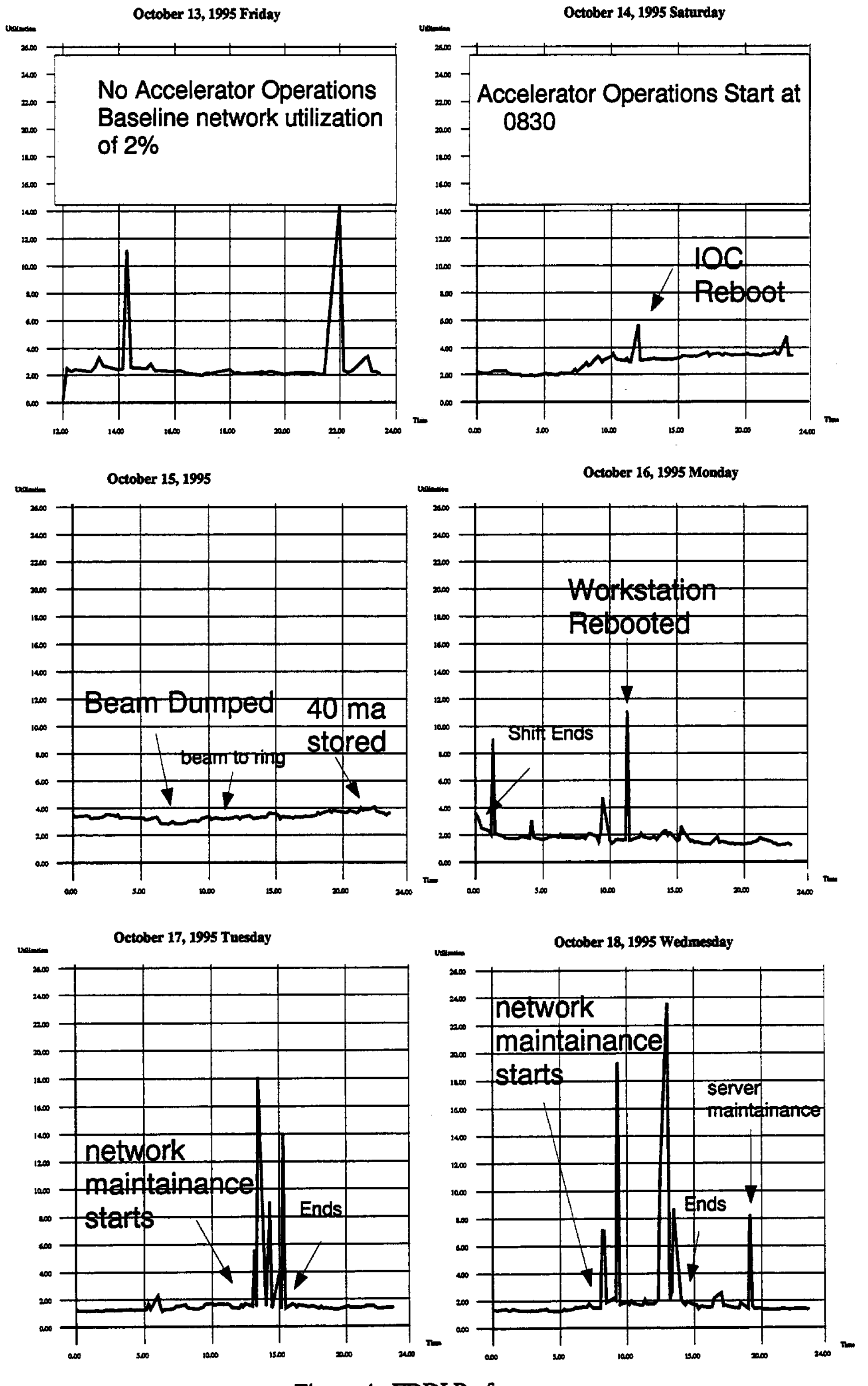


Figure 4. FDDI Performance

approximately 0800 on October 14th, the accelerator systems was brought on-line. At 1803 the linac e-gun was turned on. During this time network utilization increased from approximately 2% of capacity to 4% of capacity and remained at 4% until the shutdown at midnight October 15th. The accelerator was not running from October 16th through October 18th. Since this was a scheduled shutdown, the subsystem groups, including controls, were testing systems during the day shift. Network maintenance and reconfiguration was performed during this time. In order to stress test the network, an EPICS server IOC and an EPICS display manager client (MEDM) were connected. The number of events per second was recorded; these are a measure of network and IOC capacity. We have noticed differences in the maximum rate attainable when the IOC and workstation are on the same Ethernet segment compared with when the IOC and workstation are connected to different Ethernet segments which are connected by the controls FDDI and the hubs. There seems to be a degradation in performance when connecting through the hub. The cause of this performance degradation will be investigated further and reported at the next opportunity.

CONCLUSION

The APS controls network has met all requirements for a high availability, minimum latency system and has been a significant help in supporting the commissioning activities. There are a few areas where improvements in performance are still being investigated and these will be reported at a later date.

ACKNOWLEDGEMENT

This work was supported by the U. S. Department of Energy, Office of Basic Energy Sciences, under contract No. W-31-109-ENG-38.

Panel Discussion on EPICS

W. McDowell
Argonne National Laboratory

Session M3B, a Panel Discussion on the Experimental Physics and Industrial Controls System or EPICS, took place the afternoon of Monday, October 30. This session was assembled due to the widespread interest in EPICS in the controls community and the use of EPICS around the world. The object was to present a balanced look at this popular product. To meet this criterion the panel selection was made to include a variety of points of view. There were EPICS developers, EPICS users, a commercial vendor of EPICS and an independent reviewer of control systems. The panel members were:

Matthias Clausen, DESY
Robert Dalesio, LANL
Martin Kraimer, ANL
Steven Lewis, LBL
William McDowell, ANL (Chairman)
Peregrine McGehee, Canada-France-Hawaii Telescope Corporation
Tim Mooney, ANL
Mark Rivers, U of Chicago
Arun Sheth, Kinetic Systems
Jiri Navratil, CTU
William Watson, CEBAF

The format of the session was as follows:

- A brief Introduction (McDowell).
- A 15 Minute overview of EPICS (Dalesio).
- A 5 to 10 Minute talk from each of the panel members.
- Open Discussion with the conference attendees.

The presentations can be summarized as follows:

EPICS is a set of software tools and applications originally developed by Argonne National Laboratory and Los Alamos National Laboratory for the purpose of building distributed control systems to operate devices such as Particle Accelerators, Large Physics Experiments, Telescopes, etc. Present and future development is being done cooperatively by Argonne (ANL), Los Alamos National Laboratory (LANL), Lawrence Berkeley National Laboratory (LBNL), the Continuous Electron Beam Accelerator Facility (CEBAF), DESY (Deutsches Elektronen-Synchrotron) and the Telescope community.

An appropriate summary of the discussion is that the purposes of EPICS are:

- To provide a fast, easy interface to supervisory control, steady-state control and data acquisition through a table-entry database.
- To provide an Operator Interface (OPI) to all control system parameters through an interactive display editor.
- To provide a means of logging data through a table-entry archiving file. To allow for the management of alarms through a table-entry alarm file.
- To provide sequential control through a state definition language that has convenient routines for database interface.
- To allow data acquisition (channel-access) routines that interface the control system data with data analysis, adaptive control algorithms and any other functions not provided in the control system.

To learn about EPICS, information can be found at:

http://www.aps.anl.gov/asd/controls/epics_home.html

The New Generation of PowerPC VMEbus Front End Computers for the CERN SPS and LEP Accelerators Control System

A. Bland, P.Charrue, F. Ghinet, P.Ribeiro, M. Vanden Eynden
SPS and LEP Controls Group
European Laboratory for Particle Physics
CH1211 Geneva 23 - Switzerland

ABSTRACT

The CERN SPS and LEP PowerPC project is aimed at introducing a new generation of PowerPC VMEbus processor modules running the LynxOS real-time operating system. This new generation of front-end computers using the state-of-the-art microprocessor technology will first replace the obsolete XENIX PC based systems (about 140 installations) successfully used since 1988 to control the LEP accelerator. The major issues addressed in the scope of this large scale project are the technical specification for the new PowerPC technology, the re-engineering aspects, the interfaces with other CERN-wide projects, and the set up of a development environment. This project also offers support for other major SPS and LEP projects interested in the PowerPC microprocessor technology.

INTRODUCTION

The mission of the European Laboratory for Particle Physics is pure science - particle physics - probing the innermost constituents of matter to find out how our world and the whole of the Universe works. To do this, it operates a number of particle accelerators, the largest of which is the LEP electron-positron collider. Beams of electrons and their antimatter counter-parts, positrons, are circulating in a 27 kilometer underground ring at nearly the speed of light and collided together. A chain of accelerators, including the Super Proton Synchrotron (SPS), provides beams for LEP. The LEP machine consists of various equipment installed in the 27 km ring tunnel and in 8 surface buildings, their associated pits and underground areas. The SPS and LEP Control System extends from a central control room to all the surface buildings and underground areas in order to provide command and acquisition facilities for the collider and the particle beams.

THE CURRENT SPS AND LEP CONTROL SYSTEM ARCHITECTURE.

The architecture of the SPS and LEP control system is modeled on three layers [1] [2]: the Control Room Layer, the Front End Computing Layer, and the Equipment Control Layer.

The Control Room Layer

The Control Room Layer is composed of modern UNIX workstations, servers and X-Terminals. Human Computer Interfaces are developed using an X-Window and OSF/Motif User Interface Management System. Several other UNIX servers are used for file storage, for public displays, for the management of alarms and for an ORACLE on-line relational database. The current computer installation for this layer is the following :

- 30 Hewlett Packard 700 series workstations
- ~100 X-Terminals
- 7 DEC workstations

The Front End Layer

The Front End Computing Layer consists of Front End process computers (FEs) based on PCs and VMEbus crates. Their main function is to provide a uniform interface to the equipment as seen from the workstations and act as data concentrators for equipment interfaced via various fieldbuses. The task assignment between FEs is made on geographical or functional criteria. The communication between the Human Computer Interfaces (HCI) running in the SPS and LEP control room and the FEs is achieved through Remote Procedure Calls (RPC) or direct TCP/IP socket connections. The current computer installation for this second layer is as follows:

- 150 OS/9 VMEbus Systems
- 110 LynxOS 2.2.1 80486 PC systems
- 45 LynxOS 2.1.0 VMEbus 680x0 systems
- 150 XENIX 2.3.1 80386 PC systems

The Equipment Layer

The Equipment Control Layer consists of Equipment Control Assemblies (ECAs) connected to the FEs via various equipment fieldbuses (1553, GPIB, BITBUS, JBUS) or via RS232/422 links. The ECAs range from G64 6809 systems running FLEX to 3U and 6U VME 68k systems running OS-9. The accelerator equipment is distributed in underground halls and in surface buildings.

The SPS-LEP Network.

Network communication is made by local Ethernet segments bridged to large Token-Rings, one for the LEP general services linking all the surface buildings and others for the accelerators. These Token-Rings have been partially replaced by 100 Mbit FDDI backbones which will eventually cover the entire CERN site. The data distribution is based on the TCP/IP protocol suite and the network management is achieved with SNMP.

CURRENT LEP FRONT ENDS ARCHITECTURE

The LEP front ends were installed in 1988 to control most of the LEP accelerator services (i.e. vacuum system, tunnel cooling and ventilation, electricity, water distribution, magnets, and so forth). They are Olivetti 386-16Mhz PCs running SCO-XENIX v2.3.1. They are disk based, i.e. each has the complete XENIX installation on its 130 Mb disk. They are connected to the network via a Token-Ring card and have a CERN-made 1553 connection to control the 1553 ECAs. The XENIX software drivers for Token Ring and 1553 have been developed at CERN. These computers run on a 24h/day and 365 day/year basis

These front ends run well now since we no longer add or modify software and/or hardware on a large scale. But as we did not follow system updates and we now have TCP/IP and NFS on our major platforms, we definitively need to upgrade these front ends. The Olivetti 386-16Mhz PCs are not available anymore due to their obsolescence and we are organizing the maintenance via a stock of spares. Our main hardware problems were local disk failures, so we renewed many of the disks in 1992. Now we start to experience Power Supply failures. On the software side, the distribution of system files to every hard disk requires complex management. In addition, the lack of standard TCP/IP and NFS now makes us reluctant to develop any further on these platforms.

PROJECT MOTIVATION

The first motivation is that we want to use the VMEbus standard for the LEP Front Ends and to use a VME CPU card based on a powerful processor. The main interests of the VME standard are its big and flexible I/O address space and its good interrupt capabilities. The second important motivation is the desire to use the Network File System (NFS) and the LynxOS operating system in all our FEs to simplify our maintenance effort, since today several generic software packages, like the one used for Remote Procedure Call, have to be maintained on too many different operating systems. But the interest is not only limited to these aspects; several major rejuvenation projects dealing with the SPS power converters (ROCS project) and the LEP Beam Synchronization Timing (BST project) have high performance requirements and will benefit from the new proposed technology.

The **two objectives** of the project are :

- the replacement of the operational PC XENIX systems (70 installations) by VMEbus systems running the LynxOS operating system. The old PC XENIX console functionality will be replaced by already-installed X-Terminals connected to the Hewlett Packard workstations.
- the setup of a development environment for the SPS and LEP VMEbus users. This environment shall provide the necessary tools to migrate the existing PC XENIX operational software (more than 100 software modules) and to perform low level development at the operating system level (i.e. drivers and libraries).

MARKET SURVEY AND TENDERING

The choice of the new VMEbus platform was made in two stages following the CERN purchasing rules. The first step was to make a large market survey to get enough information on what was available in industry to write and submit an official tender for the supply of a VMEbus processor module.

The market survey [3] was designed to detect which was the most appropriate VMEbus processor card running LynxOS with an ethernet connection. The result of this preliminary enquiry was used to write the official invitation to tender for the supply of these VMEbus CPU boards. The market survey was sent to 82 companies in 12 countries in June 1994. We received 22 replies from 5 countries in August 1994 which were proposing 22 different CPU boards. The most popular processor proposed in these commercial offers was the PowerPC family: 2 proposals for the 601, 6 for the 603 and 3 for 604. As we felt that the PowerPC family [4] and in particular the 603/604 processors will be positioned as the high performance successors to the 680x0 family, we decided to focus the invitation to tender on the PowerPC technology.

The invitation to tender [5] was sent to 12 companies from 5 countries in October 1994. It stated that CERN asks for a PowerPC processor implemented in the VMEbus Standard running the LynxOS Real-Time Operating System. The processor module must be a manufacturer's standard product nearest to the specifications. The main requirements that were laid down for the Processor module were:

- PowerPC 603 easily upgradable to 604.
- 32Mbyte of RAM
- 512Kbytes of flash EPROM.
- 32-bit DMA interface to standard Ethernet IEEE 802.3.
- A16-A32/D8-D64 bit wide VMEbus interface.
- One VMEbus slot wide for up to 64Mbytes DRAM memory
- One or two PCI mezzanine slots.
- Standard LynxOS from LynxRTS for execution in diskless and network bootable.
- TCP/IP, NFS plus standard LynxOS drivers (VMEbus, SCSI, Ramdisk, ...).

The tendering exercise resulted in 8 replies. The successful bidder was the CES company [6] of Geneva, Switzerland, and the VME CPUboard proposed was the RIO2-8060 based on a 603-64Mhz with 32MB DRAM, 4MB Flash Eprom, 10baseT ethernet connection and 2 PCI mezzanine slots. The datasheet of this board specified 65 SPECint92 and 60 SPECfp92 for a 603/64Mhz and 192 SPECint92 and 198 SPECfp92 for a 604/96Mhz.

NEW FRONT END ARCHITECTURE

The new generation of FEs will consist of a standard VMEbus crate, a VMEbus processor module based on the PowerPC family, various fieldbus controller modules and equipment dedicated I/O modules. LynxOS is the chosen operating system for the new FEs.

The LynxOS installation will also include :

- host-based TCP/IP networking software for Ethernet;
- Network File System (NFS), client and server;
- NetBoot firmware for bootstrap loading LynxOS systems over the network;
- LynxOS drivers for VMEbus, Ethernet, SCSI-II, and RAMDISK.

The connection of the FEs to the local Ethernet segment will be made directly from the processor module. Connections to the future FDDI network will be made via a separate Data-Link module tightly coupled to the processor via a dedicated PCI interface card, conforming to the PMC specification.

Software Development Environment

As the new generation of FEs will be diskless, bootstrap load files are prepared on development systems, stored on a boot server and loaded into target systems over the network. Application programs will be compiled, linked and debugged with the GNU software (gcc, g++, gdb). This development activity will be possible either from PowerPC LynxOS native development systems, or by using cross-development facilities running on an IBM AIX workstation. This second option offers to the developers the comfort and performance of a classical workstation development environment. The binaries produced will then be stored on a NFS file server from where they will be loaded onto the target systems.

PROJECT MANAGEMENT

Preliminary Considerations

At the beginning of the project it was decided to use mechanisms to monitor progress, generate milestones and reduce the risk linked to the technology involved in the project. Various aspects such as:

- the project scale (140 XENIX systems)
- the new PowerPC technology
- the different categories of users (application programs developers but also low level operating system specialists)
- the interface with major projects like ROCS and BST
- the re-engineering aspects
- the operational constraints (continuous LEP accelerator operation required)

led us to use software and hardware engineering standards to manage the project. The IEEE software engineering standards [7] and the European Space Agency (ESA) PPS-05 standards [8] were used to address both technical and managerial project issues.

Project lifecycle

The project organizational structure, the organizational boundaries and interfaces and the individual responsibilities are specified in the project management plan (PMP) document. The project lifecycle is divided in six phases : the user requirements phase, the system requirements phase, the architectural design phase, the detailed design phase, the transfer into operation phase, and the operation and maintenance phase.

USER REQUIREMENTS PHASE

This is probably the most crucial phase of the entire project. The objectives of this phase are to identify all entities involved in the project (this activity was mandatory due to the lack of documentation concerning the existing software modules) and capture the requirements for the future operational and development systems. Several technological evolutions (i.e. cross-development environments) as well as the experience gained during the past seven years with the XENIX systems raised new user requirements. After being published, the User Requirements Document (URD) was reviewed and accepted as the baseline
document for the next project phases.

PROJECT PROTOTYPING ACTIVITY

According to the technical issues involved in the project, it was decided during the system requirements phase of the project to spend time on prototyping. As shown in figure 1, this prototyping activity is aimed at :

- improving the definition of the user requirements. Building a prototype may lead to new requirements or to the discovery of incomplete or contradictory requirements;
- gaining knowledge about the technical feasibility, the system requirements, and any aspect related to the operation of such a system;
- producing a model for the final development and operational systems.

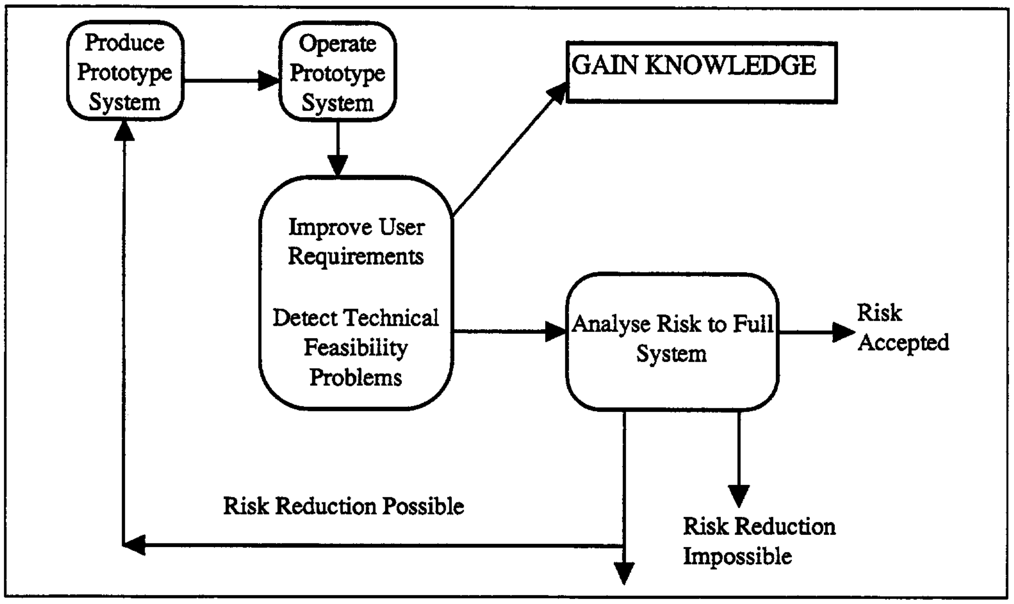

Produce Prototype System
Operate Prototype System
GAIN KNOWLEDGE
Improve User Requirements
Detect Technical Feasibility Problems
Analyse Risk to Full System
Risk Accepted
Risk Reduction Possible
Risk Reduction Impossible

Figure 1: Project Prototyping Phase.

PROJECT DESIGN PHASES

The architectural and detailed design phases are aimed at defining a collection of software and hardware components and their interfaces and building the final system. The installation of the development environment, the writing of the System User Manual (SUM) and the migration of the XENIX software modules will take place during the detailed design phase.

CONCLUSIONS

At present we are tackling the prototyping phase of the project. The acceptance tests for the first PowerPC 603 processor board have been successful and the bulk delivery will start before the end of 1995 and will continue during the first quarter 1996. The prototyping activities will be developed in two directions: several target systems will be installed in parallel with the existing XENIX PCs to check their behavior under operational conditions and, at the same time, developers will try to migrate some of the XENIX software modules, using either the new cross-development environment or native LynxOS development systems, depending on the nature of that software.

The architectural and detailed design phases will start before December 1995 and the timetable for the transfer into operation has still to be negotiated.

The new FEs are expected to provide a gain by a factor of ten to twenty in computing power compared with the old XENIX FEs.

REFERENCES

[1] SPS and LEP Controls, Status and Evolution Towards the LHC Era - This conference - R. Lauckner, R. Rausch - CERN

[2] Interfacing Industrial Equipment to CERN's Accelerators and Services Control System - P.Anderssen, P.Charrue, R.Lauckner, P.Lienard, R.Rausch, M.Tyrrell, M.Vanden Eynden

[3] Market Survey for the supply of processors in VMEbus standard with LynxOS real-time operating system software - CERN MS-2285/SL

[4] PowerPC News home page - http://power.globalnews.com/ppchome.htm

[5] Specification for the Supply of PowerPC VMEbus Processor Modules Running LynxOS Real-Time Operating System Software - CERN IT-2285/SL - SL/Tech.Spec.94-05-CO

[6] Welcome to CES 's WWW Server - http://www.ces.ch/CES_info/Welcome.html

[7] Software Engineering - IEEE Standards Collection - Institute of Electrical and Electronics Engineers, Inc

[8] Software Engineering Standards - Prentice Hall - C.Mazza, J.Fairclough, B.Melton, D.De Pablo, A.Scheffer, R.Stevens

INTERFACING TO ACCELERATOR INSTRUMENTATION*

T. J. Shea
Brookhaven National Laboratory
Upton, NY 11973

Abstract

As the sensory system for an accelerator, the beam instrumentation provides a tremendous amount of diagnostic information. Access to this information can vary from periodic spot checks by operators to high bandwidth data acquisition during studies. In this paper, example applications will illustrate the requirements on interfaces between the control system and the instrumentation hardware. A survey of the major accelerator facilities will identify the most popular interface standards. The impact of developments such as isochronous protocols and embedded digital signal processing will also be discussed.

I. INTRODUCTION

Survey of Instrumentation Groups

In September and October of 1995, instrumentation personnel representing 21 facilities and projects were contacted via email and asked to answer several questions about instrumentation interface technologies. The results are concisely summarized as follows:

- Less than 10% have built a GPIB instrument
- About 20% currently include VXI based instruments and nearly 50% plan to use VXI in the future
- nearly 60% use LabVIEW for bench measurements
- About 40% use portions of the control system during system development

In addition, their other comments were used to select the topics covered in this paper.

Accelerator Measurements

A very simple measurement setup is illustrated in Figure 1 with the accelerator as the device under test. The stimulus might drive a broadband kicker, the quad bus, or a steering magnet. The response might be measured by a wide band position monitor, a tune meter, or an orbit monitor. In any case, a correlation plot is produced. Since time is the implicit free variable, the measurement equipment is almost always synchronized with the accelerator timing system. If automatic control of the measured parameter is desirable, the measurement system could be converted to a feedback system.

An accelerator instrumentation system provides the response measurement. Figure 2 illustrates the signal flow through a generic instrumentation system. For continuous data streams like those produced by storage ring and CW linac instrumentation, buffer size is determined by the worst-case latency of the interface. If this data is used in a feedback loop, the performance of the loop is limited by the loop delay as determined by the total worst-case latency. Therefore, the interface should provide deterministic latency as well as adequate throughput. An example of low latency interface is the VME-based reflective memory used for an orbit feedback test at SPEAR [1]. Although the sampling rate for this experiment was 37 Hz, the goal for a similar system at the Argonne APS is to use a 4 kHz sampling rate and provide a - 3 dB closed orbit correction bandwidth of 100 Hz [2]. Given the high performance processor and network hardware currently available, it is unfortunate that these systems must still utilize a data path that is independent of the accelerator control system.

Referring again to Figure 2, a current trend is to reduce analog complexity and digitize closer to the source. This can lead to fairly high raw data rates. If calculated parameters are desired, a local digital signal processor

* Work supported by the U.S. Department of Energy

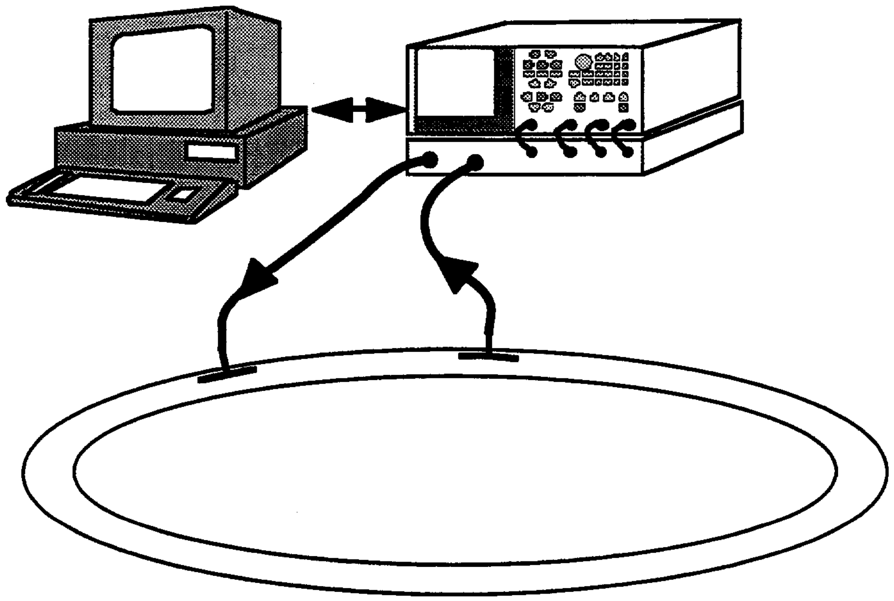


Figure 1. Simplest stimulus-response measurement

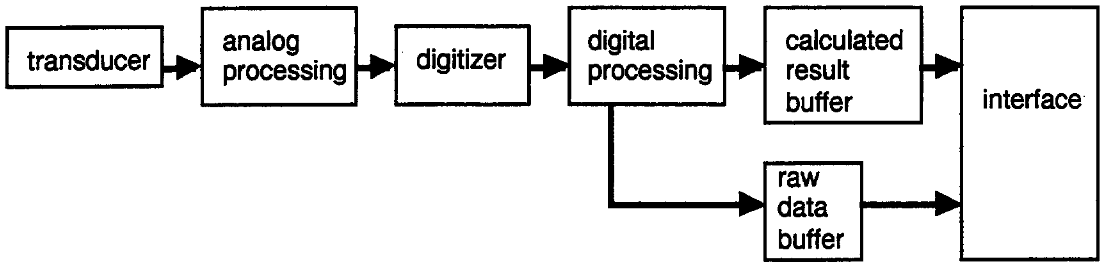


Figure 2. Generic signal flow diagram

(DSP) can write them to the result buffer at a reduced rate. An example of this simple DSP application is the low intensity position detector proposed for the fixed target areas at Fermilab [3]. Here, a 1.8 Msample/s 12 bit data stream is digitally down converted and filtered to a much lower rate, higher resolution stream that can be read through the control system interface.

II. INSTRUMENTATION SYSTEM LIFECYCLE

Development

Throughout system development, bench measurements are made using commercial test and measurement equipment. Software may be developed to fully exercise the system and obtain complete calibration data. This single user, benchtop environment is well supported by the commercial test equipment and software vendors. The accelerator control system is often immature during this phase and may lack the extensive instrument drivers, data analysis tools and the rapid turnaround environment of commercial software packages like National Instrument's LabVIEW. Therefore, it is not surprising that much instrumentation is developed without control system support and that system integration activities dominate a lengthy commissioning phase.

Even with the vast array of commercially available hardware, it is rarely possible to meet system performance and cost goals with exclusively off-the-shelf purchases. This is particularly true with systems that have high channel multiplicities. However, the in-house development timeline can be compressed by dividing a system into concurrently developed modules. Obtaining a fine grained modularity through the use of mezzanine cards allows the following benefits: chances are better that some modules might be reused in other systems, reduced module complexity improves testability, off-the-shelf modules are less likely to include expensive unwanted features, and late design changes will have a reduced schedule impact. In the Relativistic Heavy Ion Collider (RHIC) Beam Instrumentation Section, Industry Packs are used to achieve this modularity [4][5].

Commissioning and machine studies

Due to the desire to constantly improve accelerator performance, commissioning and machine studies are ongoing activities that share the following characteristics:

- unusual beam parameters and acquisition scenarios
- limited faith in system performance and reliability for these operating scenarios
- large data sets
- flexible, rapid turnaround application development tools required
- experts on hand to solve problems and provide support

Operations

During operations, required data rates are typically lower than those required by studies and some system features may not be used. It is convenient to use parameters derived locally from the raw data rather than acquire and display large data sets. Built-in self test sequences can instill confidence in system performance without requiring the additional test equipment that was used during the development phase.

Upgrades

As a result of operational experience, some systems may require improved performance. The upgrade of the LEP narrow-band orbit measurement electronics is just one example [6]. At the same facility, a new operating mode using bunch trains impactson nearly all instrumentation systems [7]. The same modularity that was beneficial during the design phase can provide similar benefits in these situations.

III. STANDARDS FROM THE TEST AND MEASUREMENT INDUSTRY.

As opposed to the industrial instrumentation and controls products, products from test and measurement companies are most similar to accelerator instrumentation electronics. However, their target market is predominantly benchtop test systems and not distributed instrumentation. The standards described in this section are used extensively during the development phase of instrumentation systems. Commercial instruments based on these standards also show up in the operational environment when their performance and flexibility justifies the typically high capital cost.

GPIB

To this day, virtually all rack and stack instruments sport an IEEE 488 port. With it's 8 data lines, 8 control signals, and bulky connector, this port looks fairly archaic, and the following timeline shows the reason.

1965 - Hewlett Packard defines its HP-IB interface for instruments

1975 - IEEE 488 standard is ratified. Defines physical interface based on HP-IB

1987 - IEEE 488.1 is ratified [8]. Evolutionary step from 1975 standard. Commonly called GPIB.

1987 - IEEE 488.2 is ratified [9]. Defines higher level command structure. Revised in 1992.

Because the 1975 standard did not define status reporting, data formats and other software protocols, manufacturers each invented their own implementations. The resulting mess is partially responsible for GPIB's poor reputation with system integrators. IEEE 488.2 defined the command structures that all instruments and controllers should support. In 1990, the Standard Commands for Programmable Instruments (SCPI) Consortium defined an extensible set of instrument specific commands.

Although some instruments allow for binary data transfers, all of the standard commands are ASCII that must be parsed in real time. Thus, a one shot request for a small data block incurs significant latency and overhead. The typical raw transfer rate for a GPIB connection is over 1 MByte/s for long blocks. The bus can have up to 15 devices, each separated by about 1 meter with a total electrical bus length of 15 meters.

As an example of real world system performance, a recent test at Fermilab was performed to evaluate oscilloscopes. The test consisted of digitizing a series of 8 bit samples at 2 Gsamples/s and reading them in through GPIB to a LabVIEW application running on a Macintosh Quadra. With the fastest scope, a 2000 point series was digitized and transferred 10 times in 270 milliseconds. With a 100,000 point series, the same scope digitized and transferred 100 times in 59 seconds.

VXI

VME Extensions for Instrumentation (VXI) is now the card/backplane standard for the test and measurement industry. The VME Handbook [10] provides a very readable overview complete with comparisons to VME. A brief summary is included here.

There are four VXI card sizes, but by far the most popular is the C-size card (6U high by 340mm deep) so this will be the assumed format in the following discussion. To provide room for electromagnetic shielding, the cards are spaced on 1.2 inch centers rather than the 0.8 inch centers defined by VME. P1 and P2 carry the bus signals defined by revision C.1 of the VME specification. The outer two rows of P2 carry the following signals:

- 10 MHz clock
- Bussed TTL and ECL triggers
- 12 pin local bus
- Analog summing bus
- Module Identification bus
- Additional power distribution

The Module ID bus, along with configuration registers defined in A16 space, allow some level of system self-initialization. Two broad classifications of VXI instruments can be defined by their adherence to VXI software standards:

1. Message Based: Usually produced by a test and measurement company that ports its GPIB instruments to VXI. Typically they will use a word-serial protocol similar to GPIB SCPI commands. The company's customer base is usually downsizing a rack and stack test system and also increasing throughput. For example, converting an oscilloscope-based digitizing system to VXI will result in a 1 to 3 increase in channel density [11].
2. Register Based: Usually produced by a data acquisition company that moves functionality from CAMAC, VME, etc., to the VXI format. Register-based devices keep the efficient memory-mapped access to data and control/status registers. To an accelerator control system, VXI register-based devices are accessed like any other memory-mapped VME card. Many modern control systems provide VME-based real time systems. In these cases, the VXI address space can be transparently bridged to the VME address space. At Brookhaven, the AGS to RHIC transfer line position monitor electronics reside in a VXI crate that is transparently bridged to the control system VME crate [4].

Message based commercial equipment is not conveniently integrated with most control systems. The messaging architecture of this equipment has a long history in the single user, bench-top test environment. A modern control system provides its own messaging standard to solve the problem of distributed control over relatively slow LANs for multiple users. With its large library of instrument drivers, LabVIEW has been used as an integration tool at several laboratories. Two examples illustrate two different approaches. The Sampled Bunch Display system at Fermilab acts as a local front end for the Tevatron control system [14]. The GUI provided by LabVIEW is not present on the control consoles. A small set of parameters is provided on the network and displayed in a laboratory standard parameter page.

A tune meter at the Berkeley ALS provides another integration example. Here, LabVIEW is again communicating with a message-based single user instrument, but the interface to the control system happens at the application layer. With this access to the control system on-line database, the technique could be extended to provide a data acquisition system with the LabVIEW graphical interface and rapid turnaround development environment [15]. A system with a similar architecture is in the prototype stage at RHIC and might be used for instrumentation-related machine studies.

IV. ADDITIONAL STANDARDS

The test and measurement standards described above are currently in use at virtually every accelerator facility. However, two points were mentioned earlier:

1. In the development and upgrade phases of an instrumentation system, fine-grained modularity (below the VXI module level) can increase testability and reduce costs in the event of a change. Since Industry Packs are used for this purpose at several accelerator labs, the standard will be summarized here.
2. Register-based VXI cards can present an efficient memory-mapped interface to the control system. However, it is sometimes necessary to distribute the modules in a topology that does not allow cost effective use of VXI packaging. This could happen when the analog functions cannot fit into the VXI format, or when performance could be increased by placing the digitizers close to distributed signal sources. In either case, it is desirable to keep the efficient memory-mapped access even though the instrumentation modules are not confined to a physical backplane. A standard under consideration at RHIC is the IEEE P1394 Serial Bus [16].

Industry Pack (IP)

A summary of features:

- slightly larger than a business card
- 2 connectors for each single wide Industry Pack
- 50 pin user connector simply wired to a header
- 50 pin logic connector memory-mapped to one port of the DSP
- 16 bit synchronous logic interface for single wide packs
- option of 32 bit interface for double wide packs
- standard 8 MHz interface or preliminary 32 MHz interface
- Many available commercially
- very simple to develop in-house for special functions
- dozens of competitors, but IP seems to have the broadest market penetration
- Other market winner will surely be the PCI mezzanine standard, PMC. But it is a larger format with demanding design requirements.

Real world performance with the RHIC Beam Instrumentation Section DSP board is quite reasonable. At 8Mhz, a 5.3Mbytes/s throughput from IP to DSP is achieved. This can be doubled to 10.6Mbytes/s with a double wide industry pack. The Industry Pack specifications also specifies (preliminary) a 32 Mhz interface, which has been implemented giving a throughput of 12.8Mbytes/s or 25.6Mbytes/s for double width Industry Pack.

IEEE P1394 Serial Bus

Some comments on this standard:

- IEEE 1212 Command and Status Register (CSR) standard with a 64 bit address space
- 100, 200, or 400 Mbits/s bandwidth
- memory-mappe- like architecture is well matched to load-store architecture of current processors.
- real world performance should be comparable to VXI backplane performance. Any standard based on RS-485, Universal Serial Bus (USB), ethernet, etc. will be significantly slower.
- LAN/WAN technologies would introduce complex protocol stacks and driver software. This is a high price to pay for simply moving modules off the backplane.
- asynchronous and isochronous (guaranteed bandwidth and latency) data transfers are supported

- maximum of 4.5 meters between nodes is standard, but work is in progress to extend to much longer distances with cable upgrades and/or bridges
- cheap ($50/node in 1995)

An Industry Pack that contains a 100 Mb/s Serial Bus interface has been developed at RHIC. Several of these devices will be used to evaluate possible applications in the collider ring position monitor and loss monitor systems. Results of this study will be reported in the near future.

V. REFERENCES

[1] Y. Chung, et. al., Closed Orbit Feedback with Digital Signal Processing, Proc. of the Fourth European Particle Accelerator Conference, London (1994).

[2] Y. Chung, et. al., Digital Closed Orbit Feedback System for the Advanced Photon Source Storage Ring, this conference.

[3] H. Ma and C. Drennan, Position Monitoring of Low Intensity Beams Using A Digital Frequency Down Converter, Beam Instrumentation Workshop AIP Conference Proceedings 333 (1994).

[4] T. J. Shea, et. al., Beam Instrumentation for the RHIC Sextant Test, Proc. of the Fourth European Particle Accelerator Conference, London (1994).

[5] Industry Pack Logic Specifications Rev. 0.7.1, Greenspring Computers, Menlo Park, CA.

[6] G. Vismara, New Front End Narrow Band Electronics for the LEP Beam Orbit Measurement System, Proc. of the Fourth European Particle Accelerator Conference, London (1994).

[7] C. Bovet, The Use of LEP Beam Instrumentation with Bunch Trains, Beam Instrumentation Workshop AIP Conference Proceedings 333 (1994).

[8] IEEE 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation.

[9] IEEE 488.2-1992, IEEE Codes, Formats, Protocols, and Common Commands, and Standard Commands for Programmable Instruments.

[10] The VME Bus Handbook, 2nd Rev., VFEA International Trade Association, Scottsdale AZ, 1991.

[11] Steve Jennings, Tektronix Corporation, private communication.

[12] E. L. Barsotti, Jr., Fermilab, private communication.

[13] RHIC Design Manual, August 1993.

[14] E. L. Barsotti, Jr., A Longitudinal Bunch Monitoring System Using LabVIEW and High-Speed Oscilloscopes, Beam Instrumentation Workshop AIP Conference Proceedings 333 (1994).

[15] J. A. Hinkson, et. al., Automated Tune Measurements in the Advanced Light Source Storage Ring Using a LabVIEW Application, Proc. of the Fourth European Particle Accelerator Conference, London (1994).

[16] IEEE P1394 Draft 7.1 IEEE Standard for a High Performance Serial Bus, 1994.

Status, Experiences and Trends of Fieldbus Standardisation

Dr. Manfred Patz
Softing GmbH
Dingolfinger Straße 2, 81673 Munich, Germany

1. FIELDBUS AS PART OF THE CONTROL SYSTEM

Fieldbusses based on proprietary or standardised definitions play an important role in industrial automation. They cover the lowest layer of the communication infrastructure in the factory and connect sensors, actuators and decentralised I/O with the installed control devices such as PLCs, DCS etc. As the data exchange is the main focus, often optimised protocols are used. A good example is PROFIBUS DP, which is intended to connect decentralised peripherals to control devices. On the next higher level of communication, fieldbusses also connect control devices such as PLCs, DCS, computers, robots and NC controllers with each other. In this case, MMS-like services and objects are mostly used. Examples are the FMS concepts of Fieldbus Foundation and Profibus or SUBMMS used by World FIP.

2. BENEFITS OF USING OPEN FIELD COMMUNICATION

The benefits for end users of using open field communication systems are obvious. The most important effect is becoming independent from one supplier. As widely accepted standards give many device suppliers the chance to integrate a communication concept into their devices, a broad range of devices will be available in the market after some time. This gives the user a high degree of security in their investments.

Another aspect is that the suppliers can sell their products to a broader market in much higher quantities, so that they are in a position to offer the devices for a much lower price than proprietary solutions, and competition will enforce this price reduction.

A third aspect is the technical progress. Again, the competition in the market will enforce that new technological possibilities are integrated into the products as early as possible.

3. WHY IS STANDARDISATION IN THE FIELDBUS AREA SO TROUBLESOME?

The international standardisation of a 'fieldbus' started approximately in 1987. Before and partly in parallel several national standardisation efforts took place: namely FIP and PROFIBUS. One would expect that seven years later a reasonable standard would be ready for use, but this is not the case. Actually, the group working on this standard has exceeded 120 persons. There are normally three meetings per year of one week duration. Some of the subgroups, e. g. the application layer group, meet 6 times a year. All together, I estimate that the international community has spent more than 50 man-years for meetings relating to international fieldbus standardisations. That is only for meeting attendance, not for any preparation.

And the results are rather poor, compared with the long duration and the tremendous efforts.

These are the reasons:

1) The subject is rather complex. On one hand, the architecture of the fieldbus protocols was oriented on the classical OSI Layer structure, but had to be optimised for performance and realisation aspects. This led to a well accepted three layer architecture. On the other hand we had to learn that very special requirements must be fulfilled in this process-oriented area. It took a long time to understand these requirements.

2) Two major consortiums and several 'key players' (companies as well as persons) have been very unwilling to make compromises. This is understandable if we look at the enormous market potential of field devices, as well as the investment the PROFIBUS and FIP consortia have made. This situation has been intensified by the fact that standardisation rules are based on the idea of achieving a compromise and that it is very easy to block a decision. Only after learning that all parties will be losers did a more co-operative attitude come about.

3) The users have viewed the standardisation as a matter for the vendors. For very long they did not really ask for standardised solutions and this allowed the vendors to play the game. Only in the recent past have the users become aware of the potential benefits resulting from standardised solutions, and now they are increasing the pressure on their suppliers to offer appropriate ones.

These changes in the assessment of the situation led to a significant acceleration of work and in promising results, but as we will see, there is still a long way to go.

4. STATUS OF STANDARDISATION

4.1 National standards

In Europe there exist two fieldbus systems which have been standardised by national standardisation bodies. These are FIP, standardised by the French committee UTE, and PROFIBUS, standardised by the German committee DKE. This standard is published as DIN 19245 part 1 to 3.

The FIP standard contains definitions for the physical layer, a data link layer, an application layer optimised for buffered data transfer (MPS) and, most recently, an application layer definition for 'classical' information transfer. These definitions are called 'SubMMS'.

PROFIBUS contains definitions for thephysical layer, a data link layer, an application layer for classical information transfer, called 'FMS', and an application interface for fast information transfer between a control device such as a PLC and decentralised peripheral devices. This definition is called PROFIBUS DP (decentralised peripheral).

The PROFIBUS standard is completed by fully developed network management services and protocols.

Due to special agreements ruling the European standardisation process, not all parts of the above mentioned definitions have reached the status of fully approved formal standards but have been published as draft standards. But this fact has only minor influence on the acceptance of that part.

In Germany there exists another standard, INTERBUS-S (DIN E19258), which is viewed as a sensor/actuator bus, but is competing with PROFIBUS and FIP in the market for some application areas.

4.2 European standardisation in CENELEC 65CX

Due to the long and risky standardisation process in the IEC, the national committees within Europe agreed in principle to approve existing national standards which are recognised in the market as European standards. For this reason a committee 65CX has been established in CENELEC. As a first task, a set of national standards which fulfil certain criteria will be published as EN 50170. FIP, PROFIBUS and possibly PNET fulfil these criteria. More recently, a first vote was taken on pr 50170 by the European national committees, but even though more than 75 % of the weighted votes were been in favour, the vote failed. Currently, the comments are being resolved and the next vote will be initiated very soon. After the preparation of this European fieldbus standard, other concepts such as Interbus-5 (DIN E19258) and DIN-Meßbus (DIN 66348) will be considered for standardisation under the CENELEC umbrella.

It seems worthwhile to mention that, in addition to the activities in CENELEC 65CX, all IEC standards are voted to become European standards by the so called 'parallel voting procedure'. In general, standards approved by IEC will also be approved by the national committees in Europe and then become European standards. Existing national standards with the same technical concept and the same application scope have to be withdrawn after a certain period.

4.3 International standardisation in IEC 65C WG6

Internationally the fieldbus standardisation is handeled by the working group 6 of IEC 65C. The IEC working group always meet together with the ISP SP50, the US national committee dealing with fieldbusses. According to the complexity of the subject, the work is spit into four different 'projects': the physical layer specification, the data link layer specification, the application layer specification and system management specification. Within ISA a fifth project is studied, the so called 'user layer' specification. The projects have reached the following status:

Physical layer:
Since the end of 1993 there has existed a physical layer standard (IS) for the voltage mode, at 31.25 kbits/s, 1.0 Mbits/s, 2.5 Mbits/s, and for the current mode. Several optional physical connections such as fibre optic and radio transmission are approved as new work items or are in progress.

Data link layer:
A data link layer specification has been circulated as an 'approved committee draft for vote (AC DV)' in June 1993 but did not reach the necessary positive votes from the national committees. In the meantime, the project has updated the service document and is now working on the protocol document. Progress is slow and a new ACDV cannot be expected before the end of 1995.

Application layer:
Due to enormous efforts of the application layer project members, the document is very close to being published as ACDV. After being published as a committee draft for comment, it was judged very positively by the national committees. But due to the activities of the fieldbus foundation consortium (FF) the acceptance of this document will be doubtful. These activities will be discussed later.

System management:
The system management project, which also includes network management, was started in 1993. A first document, describing the architecture was published in 1994 for comment. However, an international standard cannot be expected before 1996.

User layer:
The user layer project is a significant approach to describing the device behaviour of devices for process control. It uses the method of 'function blocks'. The ISA user layer group has published a technical report with more than 1000 (!) pages which describes in great detail the behaviour of approx. 30 function blocks.

5. Related Activities

There are two other activities which round off the described protocol specifications.
Profiles:
Due to the very high complexity of the fieldbus specifications several so-called profiles shall be developed for factory automation and process control. These profiles shall select the required communication services and shall define the communication parameters.
Function block definitions:
In IEC 65 WG6, a general framework for function block standards is under development. In this 'metastandard' the general structure of a function block, its interaction policies, etc. are described. It is important to mention that the concepts under discussion are compatible with the language definitions in IEC 1131-3. Now, a new work item has been established to standardise the details, described in the ISA SP50 technical report, with the above mentioned function block metastandard.

6. What Happens in the Market?

6.1 Strategy of existing concepts: PROFIBUS and FIP

It became very clear in the past two or three years that the high expectations for one international standardised fieldbus which fulfils all the requirements is an illusion. Especially, the long time needed for the specification has disappointed device supplier as well as vendors. This situation is a very good breeding-ground for existing solution with well proven technology. The members of the PROFIBUS and FIP consortia took this opportunity to penetrate aggressively into the market. Especially PROFIBUS, which has had a great success and has established regional organisations in 13 countries and has more than 200,000 nodes are installed.

At that stage, three concepts were offered to the market: PROFIBUS/ISP, FIP/World FIP and the evolving IEC concept. This was obviously not acceptable to the big users, especially in the USA. They kept on pushing for a single solution.

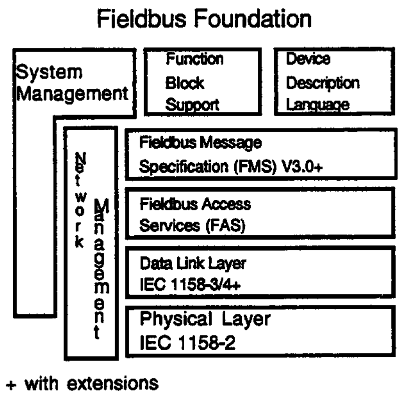
6.2 Fieldbus Foundation

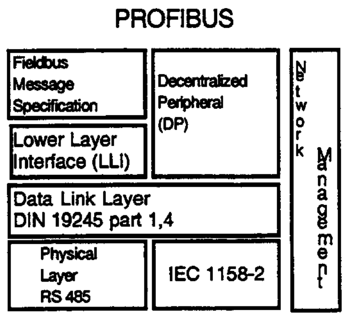
In June 1994 the merger of ISPF and world FIP North America was proposed and has been approved by the membership of both organisations. The new organisation was called the 'fieldbus foundation' (FF). Now all the key players are again in one consortium, but this fact has been bought by a significant technical change. Instead of the proven PROFIBUS data link layer, the uncompleted and untested IEC data link layer will be used. The specification work has been started, but a specification comparable to the one provided by ISP will not be available before the end of 1995. This means that the ISPF - WorldFIPNA merger has to be paid for by a delay in product availability of approximately 2 - 3 years.

7. AN ATTEMPT OF PROPHECY: THE FUTURE OF FIELDBUS

The experience in the past shows that any attempt to estimate the future of the subject 'fieldbus' needs a high degree of prophecy. In the past, no clear evolution line could be seen and too many powerful interests have influenced the progress. This is not only true for the standardisation, but also for the events in the market.
But in order to get some idea of the future some tendencies can be seen:

- The vendors realise more and more that the communication capabilities at the fieldbus level are an 'enabling' feature and not so much a competitive one. This increases the willingness to cooperate with other vendors.
- The user becomes more and more aware of the benefits of interoperable, open automation systems. The pressure to agree on one fieldbus standard is increasing. This fact led finally to the merger of ISPF and World FIP North America.
- The experience with PROFIBUS in the last few years shows very clearly that those concepts are accepted by the users and are easy to use. This aspect has priority above the basic interface prices. Under the aspect 'easy to use', concepts such as CAN and PROFIBUS-DP have very good chances.
  For the other concepts, where high functionality is accompanied by high complexity, greater effort is required to hide this complexity from the users by providing intelligent interfaces and tools.
- The future of the fieldbus business will come in evolutionary steps and not in a revolution, bringing the final solution in a big bang. There is a relatively smooth migration path from well-tested and accepted concepts like PROFIBUS and FIP to the next generation, which might be a fieldbus according to Fieldbus foundation specification. This becomes evident by comparing the protocol architecture of FF and PROFIBUS:


Fieldbus Foundation
System Management
Function Block Support
Device Description Language
Network Management
Fieldbus Message Specification (FMS) V3.0+
Fieldbus Access Services (FAS)
Data Link Layer IEC 1158-3/4+
Physical Layer IEC 1158-2
+ with extensions


PROFIBUS
Fieldbus Message Specification
Lower Layer Interface (LLI)
Decentralized Peripheral (DP)
Network Management
Data Link Layer DIN 19245 part 1,4
Physical Layer RS 485
IEC 1158-2

- Further, experience with the well established fieldbus systems, such as PROFIBUS, shows that, for end users, the details of a communication system are of very low interest. Instead, questions of usage and integration are pushed into the foreground. Therefore concepts to establish "play and play" are in discussion and in some cases under development. For example, the concept of function block and function block shell would significantly reduce the effort of the communication stack in field devices. Symmetrically, the concept of a device description language would enable control devices to handle field devices with a broader range of functionality. The basic idea is that the functionality of the devices is coded inside the field device and can be questioned by the control device. Consequently, the control device can adjust its performance by interpreting the device description of the field devices. This will make possible interchangeable devices in control structures.

8. WHAT SHALL THE USER DO?

With the merging of ISPF and world FIP NA and with the necessary restart of the specification work, the hope to get products based on an internationally accepted standard in the near future has been lost. Under optimistic estimations, a delay of two years has to be taken into account.

But this does not mean that the users have to wait again. They can use very mature systems today and do not risk loss of their investment when the next generation come to market. The future systems will use the same physical layer and provide the same or very similar user interfaces. This means that the investment in cabling and in the integration of fieldbus into the application is worthwhile.

The future systems will provide higher functionality but also higher complexity. To cope with this aspect, an early start with existing open fieldbus systems, e. g. PROFIBUS, is a wise decision. The real challenge is not the technology of a fieldbus, but the knowledge how to design, to install and to operate open, distributed fieldbus systems.
It is never to early to start with the acquisition of this knowledge.

The XC6200 FastMap^{TM} Processor Interface

Stephen Churcher, Tom Kean, and Bill Wilkie
Xilinx Inc.
(presented by Raj Patel)

Abstract

The Xilinx XC6200 is the first commercially available FPGA to be specifically designed for use within microprocessor-based systems. This paper discusses the architecture of the key element in this device: the *FastMap^{TM}* processor interface. Salient features of the user-programmable part of the XC6200 are also described.

1 Introduction

Almost all modern digital systems consist of three major functional components: microprocessors, memories and logic ICs. Logic ICs interface microprocessors to physical devices such as screens, keyboards and networks and perform computations which are unsuited to the microprocessor. Logic may be implemented using mask programmed devices or Field Programmable Gate Arrays (FPGA's) [1]. An interesting question is: What would the ideal logic part, from the point of view of a microprocessor, look like?

- Processors view the world as a sequence of memory locations. Therefore the interface to the FPGA should be through memory mapped registers.
- Processors run different programs at different times. FPGAs should therefore be able to be reconfigured for different tasks at different times.
- Since processors are synchronous devices the FPGA should be able to operate from the same clock as the processor, thereby allowing predictable interactions between them.
- The FPGA should be dense and fast and have good I/O capability.

2 Memory Mapped I/O

The concept of memory mapped I/O is simple: a particular register within the user's design selected via the processor address bus is connected to the processor data bus and a read or write operation is performed. The address selection and connection to the external bus is typically implemented using multiplexors and gates within the FPGA's user resources, although this has several disadvantages.

- The interface to a modern processor is complex and high speed and can use up large quantities of user logic and IOB resources.
- If the registers which are to be accessed from the processor are placed near the centre of the device, long routing wires are required to reach user IOBs. If many registers are to be provided, complex selection circuitry is necessary.

When one considers that the control memory of an FPGA is itself a static RAM, with address and data busses available on the device, it becomes clear that these resources could be presented off the chip as the primary programming interface rather than being hidden behind a serial channel. Naturally, this requires many more pins to support wide address and data busses, and so is not appropriate in many applications. However where the device must work with a microprocessor it is advantageous.

Register resources on the FPGA can be addressed using bit and word lines within the RAM array in the same way as configuration memory cells, and their contents presented on the external data bus. This technique was first implemented in the Algotronix CAL1024 chip [2]. Algotronix technology was acquired by Xilinx in 1993, and has been continuously developed since then resulting in the XC6200 family. Figure 1 shows the first device in the family, the XC6216.

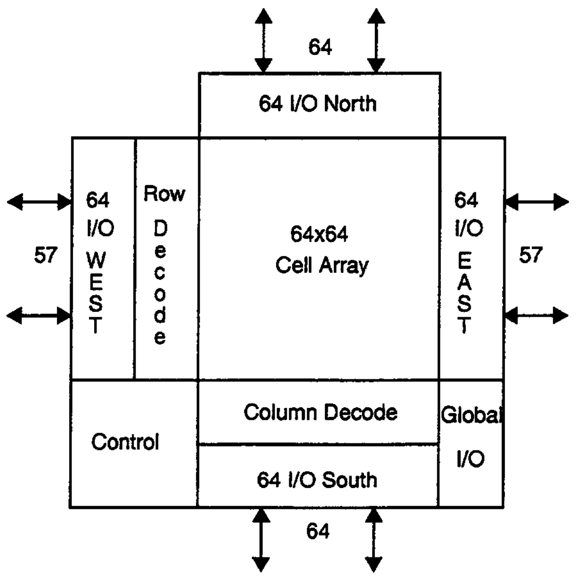


Figure 1 : XC6216 Architecture

Several additional steps are required to turn user registers, mapped into the device configuration memory, into a high bandwidth channel transferring 32 bit words between the processor and the user design on the FPGA. Firstly, the register bits, which will be physically dispersed among the configuration bits, must be collected together in the address space so that complete words of register memory are formed. This is achieved in the memory row and column decoders. Secondly, some flexibility must be provided in the mapping of register bits which are to be accessed in this way onto device cells - ideally a set of user registers located in arbitrary parts of the device could be grouped into a word of memory accessible through the processor interface.

The row and column (bit line and word line) addressing scheme used by memories makes collecting arbitrary registers on the device into a single word difficult: a realistic constraint is that all the registers should be in the same column of cells. An issue also arises concerning the manner in which bits in the processor word are mapped into registers. If full flexibility was allowed, 6 bits (to select one of 64 cell positions in a column) would be required for each of the 32 data bus bits. This is too much information to change quickly, in order to allow selection between different registers. A realistic limitation is that the bits appear in order in the processor word with the register with the lowest cell y-coordinate first. Given this constraint, the locations of the register bits can be specified with a 64 bit map register - a 0 in the map register (for example) indicates that the cell with the corresponding y-coordinate will take part in the transfer. This technique allows the registers to be spread out over the column in an arbitrary manner and can be implemented relatively efficiently in the RAM bit line logic. Figure 2 shows an example of accessing a register within user logic via an 8 bit external bus on the XC6216.

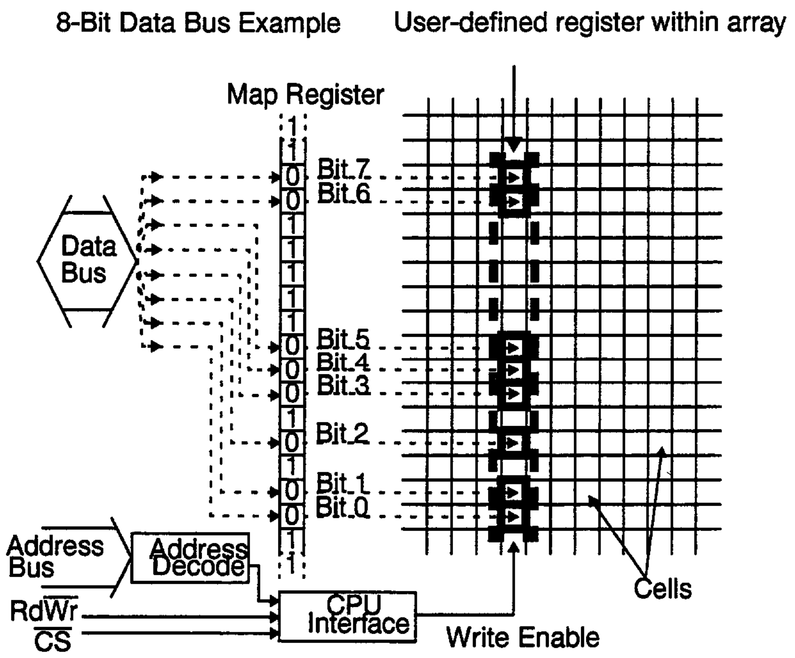


Figure 2 : Map Register - Principles of Operation

3 Dynamic Reconfiguration.

SRAM-based FPGAs are inherently capable of dynamic and partial reconfiguration; all that is required is to bring the internal RAM data and address busses onto device pins, as was done on the Algotronix CAL1024 part. The XC6200 family is configurable to provide an 8, 16, or 32 bit external data bus, and a 16 bit address bus. Using these features, the entire configuration memory can be programmed in under 100 μs.

Normally, however, it is not necessary to program the entire device. In such cases, the random access feature allows arbitrary areas of the memory to be changed. In addition, a mask register is provided which allows a subset of bits within a word to be masked out of a transfer to the memory. This is useful because often a user will wish to reconfigure a particular logical resource (e.g. a routing multiplexor) which will represent only a small fraction of the bits within a word of program memory. This feature is particularly attractive in combination with the wildcard registers discussed below.

Another property of FPGA configurations, particularly datapath-type designs, is that they are very regular, i.e. the same pattern of bits may appear at many locations in the memory (e.g. each slice in a 32 bit datapath). The XC6200 supports writing the same configuration information to multiple locations in the control memory simultaneously, using so-called 'wildcard' registers. These registers modify the row and column addresses supplied to the chip, putting 'don't cares' on corresponding address bits. For example when one address bit is subject to a 'don't care' condition, two memory locations will be written simultaneously on every transfer (Figure 3). Using the wildcard addresses, the configuration memory for all the cells on the chip can be cleared in a single memory cycle. Most user designs will use a fraction of the chip's resources, and in such cases it will often be faster to clear the configuration

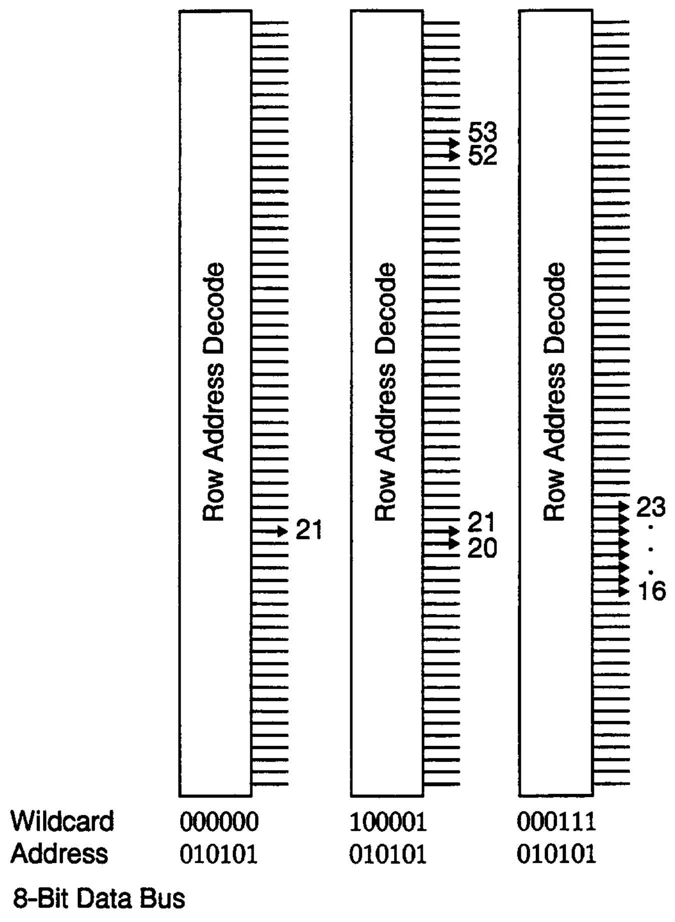


Figure 3 : Wild Card System

4 Synchronous Access.

The communication between the processor and the FPGA can function most effectively when they are both running from a common clock. This synchronizes input and output of data through the processor interface with computations running in the user logic and ensures that setup time requirements are met on write accesses to user registers and that read accesses obtain valid data. For this reason the clock signal for the processor interface is also supplied as one of four low skew global signals to the user logic [3].

As well as synchronizing FPGA and processor communication, it is useful to provide a mechanism for signalling to the user logic that a transfer to or from a register has occurred. This allows the user logic to begin processing an input value or to start computing a new output value. This function could be implemented by using a second user register as a flag to indicate transfers, but this would double the number of processor accesses required. Instead, bit and word lines used for transfers to user registers are made available as inputs to programmable routing switches within the array. By connecting these wires to logic gates within their design, users can monitor the transfers through the processor interface and take appropriate action.

5 Density and Performance.

The first member of the XC6200 family, the XC6216, contains a 64x64 array of fine grain programmable logic cells. Each of these can implement any logic function of two variables, or act as a 2:1 multiplexor. In addition to these combinatorial resources, each cell contains a register which may be used in consort with the logic function generator in a variety of ways, as shown in Figure 4. Finally, each cell in the array contains four interconnection multiplexors, which provide 'nearest neighbour' routing resources for up to four independent signals.

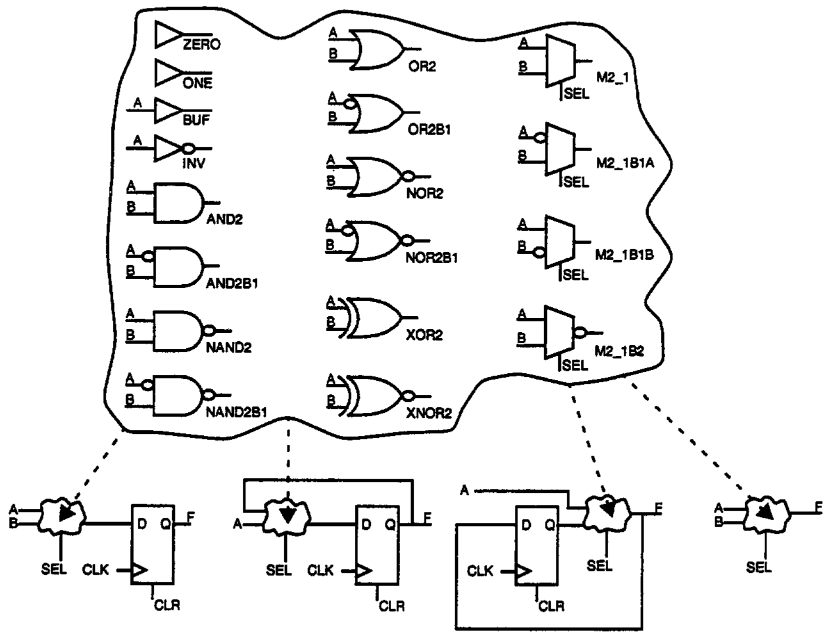


Figure 4 : XC6200 Function Unit Logic Configurations

The routing architecture of the XC6200 is depicted in Figure 5, and is a hierarchical structure with neighbour connections between cells, wires which span 4 cell blocks, wires which span 16 cell blocks and wires which cross the complete 64x64 cell array. There are also four global signals (clock, clear, and two user defined signals). These configurable connections, when combined with the 'wireless' I/O capability afforded by the *FastMap™* interface, provide users of the XC6200 with a powerful, flexible combination of routing resources.

The XC6200 family is implemented in 0.6 µm triple metal CMOS technology, and is expected to meet state-of-the-art system performance criteria. At the time of writing, full performance figures were unavailable. The equivalent gate count for the XC6216 is 16k gates for typical applications; however this figure is potentially as high as 50k gates for designs which are particularly register intensive. Smaller and larger family members will also be made available.

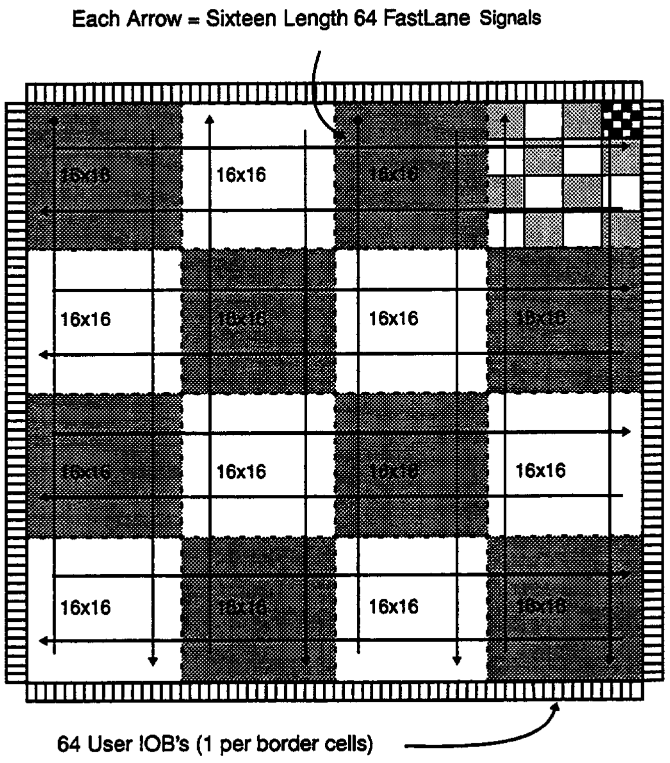

Each Arrow = Sixteen Length 64 FastLane Signals
16x16
64 User IOB's (1 per border cells)

Figure 5 : XC6216 Hierarchical Routing

6 Summary

The XC6200 family [3] is the first commercial FPGA to address the requirements of interfacing programmable logic to microprocessors. With its combination of dedicated random access parallel interface, flexible hierarchical routing, and powerful fine grain logic function generator, the XC6200 is especially suited to 'embedded processing' applications where high bandwidth peripheral devices must be interfaced to a computer system, and processing of the incoming or outgoing data stream is required.

References

1. The Programmable Logic Data Book, Xilinx Inc, San Jose CA, 1994.

2. CAL1024 Data Sheet, Algotronix Ltd., Edinburgh UK 1990.

3. Xilinx XC6200 Family Preliminary Product Description, Xilinx Inc, San Jose CA 1995.

CDEV: An Object-Oriented Class Library for Developing Device Control Applications

Jie Chen and Graham Heyes
Data Acquisition Group

Walt Akers, Danjin Wu and William A. Watson III
Control Software Group

Continuous Electron Beam Accelerator Facility
12000, Jefferson Avenue
Newport News, VA 23185

Abstract

The Control Device API (CDEV) is a highly modular and extensible object-oriented C++ class library that provides a standard interface to one or more underlying control or data acquisition packages through a common framework into which system developers can fit custom code. It defines a set of abstract classes from which a new CDEV service-layer can be developed by inheritance and accessed with the same API through run time dynamic binding. All I/O in the system is handled as synchronous or asynchronous messages to devices that may span multiple services. CDEV routes messages to appropriate services by a name service and dispatches multiple services to handle service specific I/O events. In addition, CDEV handles data transfer through a data object that may contain multiple tagged values of different types, allowing flexible I/O between clients and servers. This paper presents the design, implementation and current status of CDEV, and shows that CDEV can be a starting point to achieve the goal of sharing software in control system applications.

1 Motivation

In building a control or a data acquisition (DAQ) application, programmers usually have to conform to application program interfaces (API) provided by one or more underlying control or DAQ services such as EPICS [1] and CODA [2]. Developing a control/DAQ application is difficult since it requires detailed knowledge of the control/DAQ services such as network connection establishment, creation and synchronization of all I/O requests and data conversion handling. As control systems become more complex, the applications tend to use several services and several APIs developed at different sites at the same time. Any applications using multiple APIs will suffers from the following limitations:

- Steep Learning Curve: Many control/DAQ services have different purposes and thus have different and complicated APIs. This requires a significant effort to learn and use. For example, the channel access [3] interface has its own set of I/O operations (eg. ca_array_get (), ca_array_get_callback()), while CODA has daReadInt(), daWriteInt() and so on.
- Poor Portability: It is difficult to take a useful application from one site and API to a different site and API. On the other hand, APIs that use object-oriented features such as inheritance and dynamic binding are typically easy to port and to extend transparently [4].
- Maintenance Difficulty: It is very difficult for an application to accommodate any changes in APIs of existing services. This increases the complexity of maintaining application source code. Any new changes of API of an existing service will result in major modification of the applications.
- Event Handling Loop Dilemma: Usually all the services have their own event handling loop. This makes it difficult to decide which event loop to use in an application and how to dispatch and synchronize all I/O events from all services.

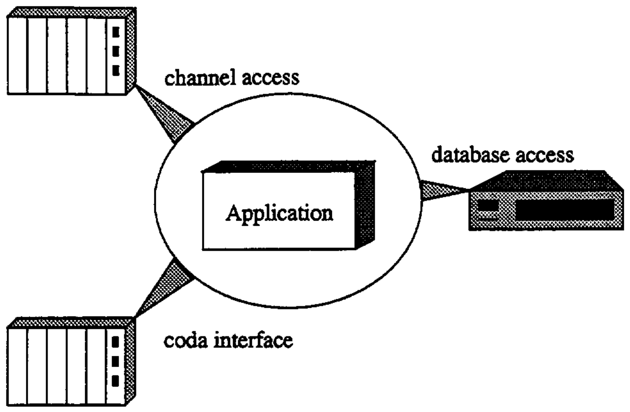


Figure 1: A typical application with multiple services

Consider a typical application shown in Figure 1. The application has to stop or start a data acquisition run according to hardware information retrieved from the channel access service and store some important data into a data base. The application not only has to handle requests to/from different services, but also has to coordinate all requests, such that no service will be blocked for a unreasonably long period of time. Furthermore, the application has to be modified or rewritten if any of the APIs or services have been changed or the application needs to access a different service.

2 C++ Solution: CDEV

It is impractical to imagine a consistently designed and all- purpose control/DAQ service which fits all the requirements of different sites. A more realistic alternative is to develop an object-oriented interface that not only encapsulates existing services, but also allows new services to be added. Due to the efficiency and wide availability of C++, it make sense to define a simple public interface for all services within inheritance hierarchies and a set of abstract C++ classes from which a new set of classes can be derived and wrapped around the API of a service. In this way, applications can use the same public interface to access different services through the derived C++ classes. We call this C++ class library CDEV.

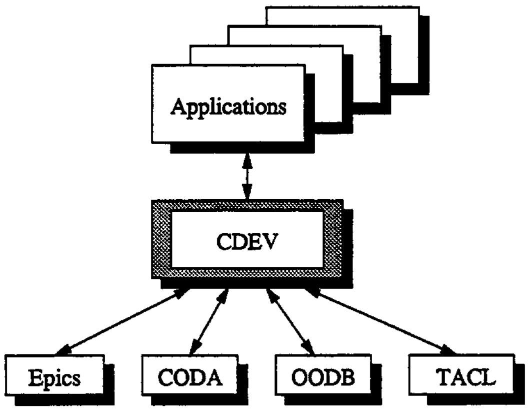


Figure 2: Architectural overview of CDEV

Figure 2 illustrates how the CDEV layer interacts with user applications and multiple services. In this approach, applications use CDEV to access these services via type-secure, object-oriented class interfaces, rather than through a set of service specific APIs. In general, CDEV is designed to reduce the complexity of control/DAQ software development without compromising performance. For example, the CDEV uses C++ language features (such as inline functions) that minimize the performance overhead of the additional layer in some of the critical sections of the CDEV code.

3 Object-Oriented Design and Implementation of CDEV

Throughout CDEV development rigorous object-oriented design is employed to ensure better quality and easier ways to communicate within the development team. This section discusses several topics related to object-oriented design and implementation of CDEV with the help of OMT (Object Modeling Technique) [5] and Booch [6] notation. To begin with, CDEV views an application as a system of *cdevSystem* which keeps information about devices, services and related resources. The system also behaves as a memory manager for all CDEV objects so that applications need not worry about memory management.

To simplify the interface to services, a device (*cdevDevice*) in CDEV is regarded as a named entity which can only respond to a set of messages such as *on*, *off* or *get/set* attributes. All I/O requests in the system are in the form of messages to devices. A message to a device can be handled either synchronously or asynchronously, and is actually carried out by a request object (*cdevRequestObject*) that is created and registered. A device will dispatch messages to a correct request object which in turn sends out to the correct control package or service. Figure 3 shows all forms of I/O requests and object relationships between a device and request objects. This message-based interface simplifies the use of CDEV and also makes it simple to put any message-based language (such as Tcl [7]) on top of CDEV. To allow for the message based interface to work with different services, a new data type *cdevData* is designed to hold different data types with different tags.

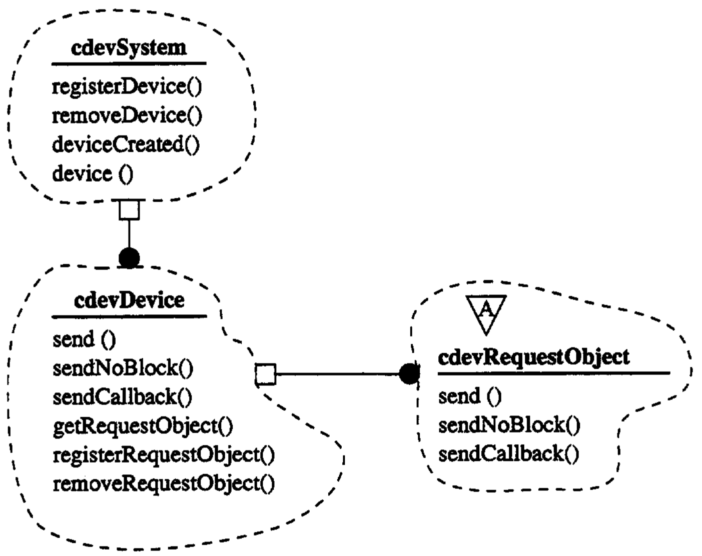


Figure 3: CDEV device and request object class category

In CDEV, a particular control/DAQ service interface is totally separated from applications and also hides the representation and internal structure of all I/O requests to/from a service. Applications using CDEV communicate with packages or services through a service layer which contains a C++ class derived from *cdevService* and a class derived from *cdevRequestObject* which handles all I/O requests. Figure 4 illustrates how CDEV uses an abstract interface for constructing a particular service of *cdevService* and how it constructs an I/O request to the services by a *cdevRequestObject*. This figure also points out the similarity between the design of this part of CDEV and the object-oriented builder pattern [8] which is widely used for a construction process that allows different representations for the object that is constructed. This object-oriented design improves modularity by encapsulating the way a complex object is constructed and represented. Applications need not know anything about the classes that define the *cdevService* or *cdevRequestObject* internal structure; such structures do not appear in the CDEV interface. Only at run-time CDEV passes messages to a name resolution system called *cdevDirectory* (see the following section) to decide which subclass of *cdevService* or which subclass of *cdevRequestObject* to construct. CDEV keeps all constructed objects to enable subsequent calls to get to the objects quickly.

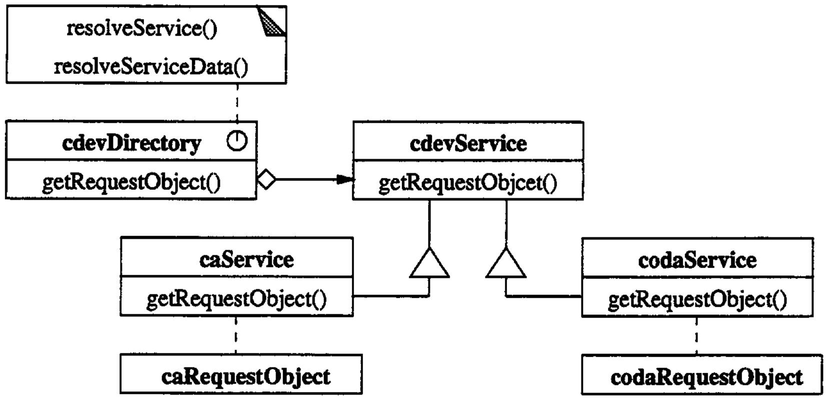


Figure 4: CDEV service and request object creation pattern

In a typical application using multiple services, one or more results due to I/O requests may arrive from different sources (such as channel access ports and CODA communication ports) and it is not desirable to block or continuously poll for incoming I/O events on any individual source. In Figure 5 a system *cdevSystem* maintains a set of services derived from *cdevSerice*. It provides methods of registering and removing these *cdevService* objects from this set at run-time. It also provides an interface for dispatching the appropriate methods of the *cdevService* objects associated with incoming I/O events. The system detects and reports the simultaneous occurrence of different types of events on multiple I/O descriptors given by the *cdevService* objects. When one or more I/O events arrive, the system of *cdevSystem* returns from the event demultiplexing call and dispatches the appropriate method(s) associated with *cdevService* subclass objects registered to handle these events. Therefore CDEV handles concurrent event demultiplexing and dispatching efficiently and frees applications from handling events explicitly.

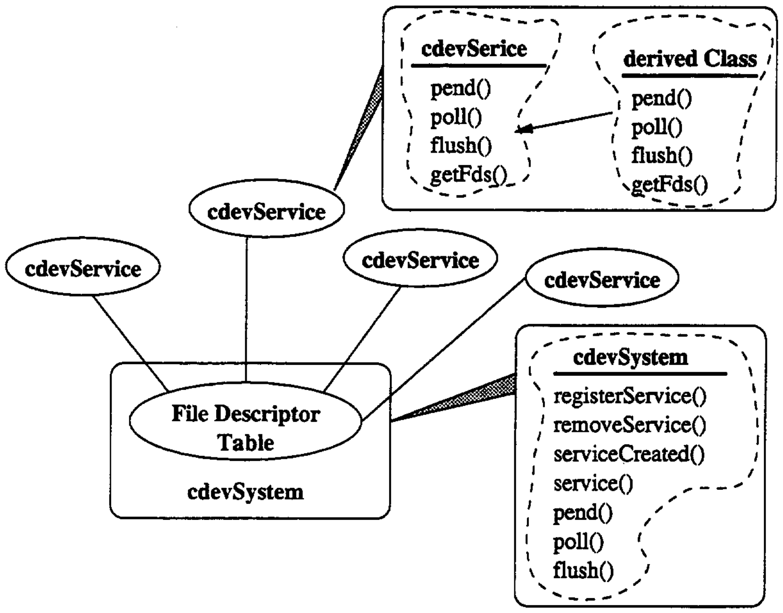


Figure 5: CDEV event dispatching mechanism

CDEV also eliminates the need for complex synchronization of multiple asynchronous I/O requests in applications. Consider the application in Figure 6, which sends out multiple asynchronous I/O requests through several objects derived from *cdevRequestObject* within a *cdevGroup* that maintains a set of *cdevTranObj* objects associated with each request object. The *cdevGroup* provides ways to add and remove these transaction objects from the group at run-time. It also dispatches the appropriate methods for the request objects and removes the associated transaction object when I/O requests have finished with notification callbacks. Therefore a *cdevGroup* object can wait for all I/O requests to finish on a set of I/O request objects. In addition, applications can use *cdevCallback* to notify itself when an asynchronous I/O request has finished.